AFOSR-TR- 91 0819

2

"OPTICAL ASSOCIATIVE PROCESSORS AND DIRECTED GRAPHS"

YEAR 1 REPORT
(1 August 1990 - 31 July 1991)

Grant AFOSR-90-0355
CMU Center Number 1-52165

Prepared by:
Professor David Casasent, Principal Investigator
Carnegie Mellon University
Center for Excellence in Optical Data Processing
Department of Electrical and Computer Engineering
Pittsburgh, PA 15213
Telephone: 412-268-2464; Fax: 412-268-6345

Prepared for:
Air Force Office of Scientific Research/NE
Bolling Air Force Base
Washington, D.C. 20332
ATTENTION: Dr. Alan Craig

1 AUGUST 1991

:LASSIFIED/UNLIMITED

ITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

:PORT SECURITY CLASSIFICATION lassified/Unlimited		1b. RESTRICTIVE MARKINGS None	
:CURITY CLASSIFICATION AUTHORITY		3. DISTRIBUTION/AVAILABILITY OF REPORT Unlimited	
ECLASSIFICATION/DOWNGRADING SCHEDULE			
RFORMING ORGANIZATION REPORT NUMBER(S)		5. MONITORING ORGANIZATION REPORT NUMBER(S) ~~Unlimited~~	
AME OF PERFORMING ORGANIZATION :negie Mellon University	6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION Air Force Office of Scientific Research	
DDRESS (City, State, and ZIP Code) 00 Forbes Avenue, ECE Department ttsburgh, PA 15213		7b. ADDRESS (City, State, and ZIP Code) Bolling Air Force Base Washington, D.C. 20332	
AME OF FUNDING/SPONSORING RGANIZATION OSR	8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER AFOSR-90-0355	
DDRESS (City, State, and ZIP Code) lling Air Force Base shington, D.C. 20332		10. SOURCE OF FUNDING NUMBERS	

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT ACCESSION NO.
61102F	2305	B1	

ITLE (Include Security Classification)

ptical Associative Processors and Directed Graphs", Unclassified

ERSONAL AUTHOR(S)
Professor David Casasent

TYPE OF REPORT Annual	13b. TIME COVERED FROM 90/8/1 TO 91/7/31	14. DATE OF REPORT (Year, Month, Day) 1991 August 1	15. PAGE COUNT 54

UPPLEMENTARY NOTATION

COSATI CODES		
FIELD	GROUP	SUB-GROUP

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
Content Addressable Associative Processors, General Memory Associative Processors, Ho-Kashyap Associative Processors, Minimum Cost Associative Processors, Pattern Recognition (cont)

BSTRACT (Continue on reverse if necessary and identify by block number)

The thrust of this research concerns associative processors. Many new aspects exist in s effort: large storage capacity, use of storage density as a measure of efficiency, use new output recollection vector encoding schemes, general 1:1 and pattern recognition y:1 associative processors, new algorithms and architectures and applications and oratory realization.

DISTRIBUTION/AVAILABILITY OF ABSTRACT ☐ UNCLASSIFIED/UNLIMITED [✓ SAME AS RPT] ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION Unclassified
NAME OF RESPONSIBLE INDIVIDUAL ~~ofessor David Casasent~~ [illegible]	22b. TELEPHONE (Include Area Code) ~~412-268-2464~~ 202-767-4931 / 22c. OFFICE SYMBOL NE

FORM 1473, 84 MAR   83 APR edition may be used until exhausted. All other editions are obsolete

SECURITY CLASSIFICATION OF THIS PAGE
Unclassified/Unlimited

"OPTICAL ASSOCIATIVE PROCESSORS AND DIRECTED GRAPHS"

YEAR 1 REPORT: 1 August 1990 - 31 July 1991
Grant AFOSR-90-0355
CMU Center Number 1-52165

ABSTRACT

The thrust of this research concerns associative processors. Many new aspects exist in this effort: large storage capacity, use of storage density as a measure of efficiency, use of new output recollection vector encoding schemes, general 1:1 and pattern recognition many:1 associative processors, new algorithms and architectures and applications and laboratory realization.

"OPTICAL ASSOCIATIVE PROCESSORS AND DIRECTED GRAPHS"

YEAR 1 REPORT: 1 August 1990 - 31 July 1991
Grant AFOSR-90-0355
CMU Center Number 1-52165

CHAPTER 1

"Introduction"

In the first year of this effort, we completed one PhD (B. Telfer) and one M.S. (K. Haines), plus 5 research papers [1-5]. The major new associative processor (AP) algorithm we use is the Ho-Kashyap (HK) AP synthesis [1]. It is *the best AP*. It has many variants for noise (Robust 1 and Robust 2 that concern two different starting matrices; Robust 2 is the best and is used extensively) and variants for storage capacity/density [we use non-unit-vector encoding for best results and refer to these as content addressable APs (CAAPs)] and variants for improved performance (minimum error cost, etc.).

Chapter 2 provides our general 1:1 memory AP results [3] (we can store more than any other AP). This also includes new CAAP concepts and our AP storage density calculations. Chapter 3 provides our many:1 pattern recognition AP results [2]. Again, we have stored far more with a much better storage capacity and storage density *than any other* AP. Our new minimum cost HK AP with quadratic decision surfaces is briefly introduced [4] in Chapter 4. This represents enormous potential and will be pursued in our Year 2 work. Our hierarchical AP results [5] are noted in Chapter 5. This is the first use of multiple APs.

"Year 2 Plans"

Our Year 2 plans are now noted. We will complete and detail (August/September 1991) our initial AP laboratory results [6]. We plan to redo these extensively with much better error source analyses and with new optical laboratory results (perfect results) suitable for analog optical or VLSI realization [7]. We hope to complete a second M.S. and formulate PhD plans beyond Year 3. We will complete documentation in a neural network journal of our closure AP and general AP work [8-9].

REFERENCES

1. B. Telfer and D. Casasent, "Ho-Kashyap Optical Associative Processors", Applied Optics, Vol. 29, pp. 1191-1202, 10 March 1990.

2. B. Telfer and D. Casasent, "Ho-Kashyap Advanced Pattern Recognition HeteroAssociative Processors", Proc. SPIE, Vol. 1347, pp. 16-32, July 1990 conference, published in January 1991.

3. D. Casasent and B. Telfer, "Ho-Kashyap CAAP 1:1 Associative Processors", Proc. SPIE, Vol. 1382, pp. 158-166, November 1990 conference, published in March 1991.

4. B. Telfer and D. Casasent, "Minimum-Cost Ho-Kashyap Associative Processor for Piecewise-Hyperspherical Classification", IJCNN'91 (*International Joint Conference on Neural Networks*), July 1991, Seattle, Washington (submitted December 1990).

5. D. Casasent and S.I. Chien, "MSE and Hierarchical Optical Associative Processor System", Proc. SPIE, Vol. 1382, pp. 304-310, November 1990.

6. D. Casasent, S. Natarajan and D. Casasent, "Multifunctional Hybrid Neural Network: Real Time Optical Laboratory Results", Proc. SPIE, Vol. 1564, July 1991 (expected publication in January 1992).

7. D. Casasent, L. Neiberg and S. Natarajan, "Analog Associative Processors: Hybrid Optical and VLSI Systems", Proc. SPIE, November 1991 conference, expected publication in March 1992.

8. B. Telfer and D. Casasent, "Neural Closure Associative Processor", to be submitted to Neural Networks.

9. D. Casasent and B. Telfer, "High Capacity Pattern Recognition Associative Processors", to be submitted to Neural Networks.

CHAPTER 2

"Ho-Kashyap CAAP 1:1 Associative Processors"

Proc. SPIE
Paper 1382-18
Volume 1382
November 1990

HO-KASHYAP CAAP 1:1 ASSOCIATIVE PROCESSORS

DAVID CASASENT and BRIAN TELFER

Carnegie Mellon University
Center for Excellence in Optical Data Processing
Department of Electrical and Computer Engineering
Pittsburgh, PA 15213

ABSTRACT

We advance new associative processor (AP) synthesis algorithms and performance measures. We compare 10 different 1:1 APs (where each input key vector is associated with a different output recollection vector). We find that unless new output recollection vector encoding techniques are used, APs are not competitive. We find the robust Ho Kashyap-2 CAAP (content addressable AP) to be preferable and that the σ_{syn} parameter used in it should not be chosen larger than necessary.

1. INTRODUCTION

APs represent a major class of neural net (NN) for noisy and partial inputs (autoassociative AAPs), for pattern recognition (heteroassociative HAPs), and to guide knowledge base queries [1-3]. We consider 1:1 APs in this paper. Many:1 APs have many different keys (distorted versions of one object class) associated with the same output recollection vector and are used in distortion-invariant pattern recognition (PR). The 1:1 APs we consider are suitable for general memory APs, AAPs, and for large class PR cases. Our results are appropriate for few:1 cases. Section 2 briefly reviews the 10 APs we consider. Section 3 notes the AP issues and performance measure we find important and our AP test procedure. Section 4 presents our data and Section 5 advances our conclusions.

2. ASSOCIATIVE PROCESSORS (APs)

Figure 1 shows the basic AP architecture we consider and Table 1 summarizes our notation. The 10 AP synthesis algorithms we consider are now briefly summarized. All are detailed more fully elsewhere [1,2]. All solve $\underline{M}\,\underline{X} = \underline{Y}$ for the AP matrix $\underline{M}$ to be satisfied for all $(\underline{x}_m, \underline{y}_m)$ where the $\underline{x}_m$ and $\underline{y}_m$ are the columns of $\underline{X}$ and $\underline{Y}$.

The direct storage nearest neighbor (DSNN) AP is the norm. Its solution is $\underline{M} = \underline{X}^T$ (the data matrix). It requires K = M which is large but not appreciably larger than for other standard APs (when K ≈ N). Its major disadvantage is that it requires winner take all (WTA) output post processing which can become computationally intensive. This is the only nearest-neighbor AP. We extended this DSNN algorithm to analog keys with the addition of an extra key vector element.

The correlation AP or vector outer product (VOP) AP solution is $\underline{M} = \underline{Y}\ \underline{X}^T$. Its disadvantage is a

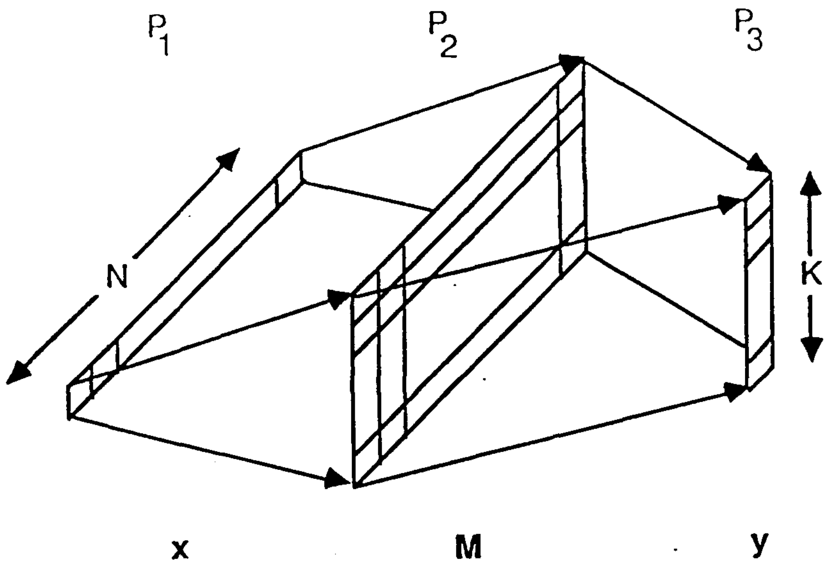


FIGURE 1: Optical matrix-vector associative processor.

NOTATION	
SYMBOL	TERM
$\underline{x}$	INPUT KEY VECTOR
$\underline{y}$	OUTPUT RECOLLECTION VECTOR
$\underline{M}$	ASSOCIATIVE MEMORY MATRIX
N	DIMENSION OF INPUT
K	DIMENSION OF OUTPUT
K×N	SIZE OF MEMORY
M	NUMBER OF STORED KEY/RECOLLECTION VECTOR PAIRS

TABLE 1: Associative processor notation.

very low storage $M \ll N$. Preferable APs are obtained using linear algebra rather than biologically motivated techniques. The pseudoinverse AP solution is $\underline{M} = \underline{M}^+ = \underline{Y}\ \underline{X}^+ = \underline{Y}(\underline{X}^T\underline{X})^{-1}\underline{X}^T$ (where the last equality is true only for linear independent key vectors). Its storage is better ($M < N$ and $M > N$) and it minimizes $J_1 = \|\underline{Y} - \underline{M}\ \underline{X}\|^2$ with respect to $\underline{M}$. The solution is exact only for linearly independent keys and its performance in noise is poor when $M \sim N$ (these are its disadvantages). The Ho-Kashyap (HK) AP is the best. It minimizes J_1 with respect to both $\underline{M}$ and $\underline{Y}$ by varying $\underline{M}$ and the recollection vector choices. It iteratively (in synthesis only) improves the $\underline{M}^+$ solution. Its $M = 2N$ storage is the largest of any AP and it handles linearly dependent key vectors (these are its two advantages) [4]. Two robust AP algorithms exist to improve noise performance with $M \sim N$. The first [5] uses $\underline{M}_1$ (with small eigenvalues omitted). The second [6] includes noise ($\underline{N}$ and σ_{syn}) in synthesis by minimizing $J_2 = E\{\|\underline{Y} - \underline{M}(\underline{X} + \underline{N})\|^2\}$ with the solution being $\underline{M} = \underline{Y}\ \underline{X}^T(\underline{X}\ \underline{X}^T + M\sigma^2_{syn}I)^{-1}$. These two robust solutions for $\underline{M}^+$ are denoted as Pseudoinverse-1 and -2 APs and HK APs using these as starting matrices are denoted as HK-1 and HK-2 APs. When non-random and non-unit recollection vector encodings are used, we refer to these as CAAPs. We use these encodings with the best APs and obtain a Pseudoinverse-2 CAAP and an HK-2 CAAP.

3. AP ISSUES AND PERFORMANCE MEASURES

We feel that analog keys and matrices are necessary to be competitive. They are also very attractive and necessary for analog VLSI and optical APs, where an analog accuracy is as easily achieved as is digital data. In practice, the keys will be feature vectors to reduce AP size (N), training set size and storage (M), and to achieve distortion-invariance. Our APs are suitable for analog VLSI and optical realization. We use linear algebra (not biologically motivated APs) to achieve storage $M > N$ (this is necessary for APs to compete). We use linear algebra to produce one pass (not iterative) APs (since they are faster and require fewer matrix-vector calculations in classification). Our algorithms are also optimum in noise and inherently allow analog accuracy (this is inherent in the algorithm, rather than ad hoc).

As performance measures we use storage density (M/NK = storage/memory size) rather than just M. We use P_C' (the percentage of recollection vectors completely correct) rather than P_C (the percentage of recollection vector elements correct). For the preferable CAAPs, we must use P_C' and in general only P_C' is meaningful. We consider only reasonable P_C' levels (90%, 95% and 99%) and we consider low $P_C' = 60\%$ levels to be useless. In our AP data, we will list the maximum M/N for which we can achieve a specified P_C'. Our data are the average of 10-100 AP runs. Other AP issues of concern in comparisons include extensive pre- and post-processing, including WTA (which we find unrealistic), and the ease of updating the APs [7].

For the 1:1 APs considered here, the keys are analog random elements uniformly distributed $[-1,1]$ and the recollection vector elements are random binary ± 1 for the standard APs. We test these APs with input noise with standard deviation σ_n and by quantizing the input, matrix, and output. To model limited accuracy inputs with δ being the separation between resolvable levels (e.g. for 16 levels (4 bits) between 0 and 1, $\delta = 1/16$), we use $\sigma_n = \delta/12^{1/2}$. For a 1% accurate processor (with 3σ 99% confidence) we use $\sigma_n = 0.003$. The $\sigma_n = 0.1$ etc. levels we use thus model noise in the input image.

4. 1:1 AP TEST RESULTS

4.1 STORAGE DENSITY (THEORETICAL)

Table 2 lists the theoretical storage capacity and density for the 6 major APs as $N \rightarrow \infty$. The storage density is quantified for $N = 50$. As seen, the correlation AP is the worse, the DSNN is comparable to the pseudoinverse and the HK AP is twice as good. Only the CAAPs provide appreciably better performance than the DSNN. The CAAP figures assume binary encoded recollections (*we consider other encoding types in Section 4.4*).

4.2 PERFORMANCE COMPARISON

Table 3 lists the maximum M/N obtained in laboratory tests for $P_C' = 95\%$ for eight APs with $N = K = 50$ with various amounts of noise σ_n present. The DSNN used $M = K = N$ and $M = K = 2N$. The robust APs used $\sigma_{syn} = 0.1$ in synthesis. As seen, the DSNN performs best (largest M = N or 2N independent of noise). Table 4 lists the storage density of these 8 APs. As seen, the DSNN is still best. This clearly indicates that standard APs cannot compete with the DSNN. We note that DSNN storage of M = 20N was obtained with no degradation in noise. The large WTA required is a disadvantage of the DSNN, but it is the only nearest neighbor AP and its storage and storage density and noise performance are best.

For completeness, we discuss these AP data, ignoring the DSNN. We showed earlier [4] that the non-robust APs have poor noise performance near $M \simeq N$ and that the robust-1 APs overcome this problem and maintain high P_C for $M > N$. The correlation AP is the worst and the HK AP is the best. The standard HK performs better than the pseudoinverse. The robust algorithms perform better for σ_n = 0.1 (since σ_{syn} = 0.1) and worse with no noise exact keys (as expected). When the pseudoinverse and HK AP performance are the same, this occurs due to the low M/N for which the pseudoinverse solution gave P_C' = 100% for exact keys and hence no HK iterations occurred. In our CAAPs, the HK improvement will be more noticeable. In the many:1 APs that we are currently developing, the HK improvement is even more noticeable.

4.3 CAAPs

Table 5 lists the maximum M/N for $P_C' > 95\%$ for binary encoded recollection vectors with K = 7 (since $2^7 > 2N = 100$). All APs (except the DSNN with M = K) have a 7×50 element matrix and a storage density (M/N)/7. Table 6 lists the storage density of these APs. These data show clearly that only with different recollection vector encodings can APs outperform the DSNN. These data also show more clearly the advantage of the different APs. The robust HK-2 CAAP is always preferable (it will never be worse than the robust pseudoinverse-2 CAAP).

4.4 CAAP ENCODING

For the HK-2 AP (the best AP), we consider four different recollection vector encodings: binary, and two standard error correcting techniques (Hamming and BCH [8]), and L-max [9]. All use 7 data bits. Binary uses K = 7, Hamming uses K = 11 (4 bits to allow error correction of 1 data bit) and BCH uses K = 15 (8 bits to correct 2 data bit errors). L-max encoding uses L-ones in a K-bit output; the two cases we consider are (L=2, K=14) and (L=3, K=9). Table 7 shows the maximum M/N for $P_C' >$ 95% for each case with different levels of noise σ_n. The σ_n = 0 performance is not of interest. The σ_n = 0.1 = σ_{syn} data is easily compared. Comparing the binary to Hamming and BCH data, we see the improvement offered by standard error-correction. We note that the L-max performance is the best or nearly so. We find L-max encoding to be preferable because ot its high-storage capacity and because Hamming and BCH require additional post processing (Hamming requires an extra matrix-vector multiplication and BCH requires an extra matrix-vector multiplication followed by an iterative procedure). L-max encoding works well because it produces sparse recollection vectors. It requires WTA detection of the L largest output recollection vector elements (this is acceptable with the low K used). We choose L-max (L=2, K=14) since the 50% improvement in M/N for doubling K is acceptaole (with the low K used). With L-max encoding we can represent

$$\begin{bmatrix} K \\ L \end{bmatrix} = \frac{K!}{(K-L)!L!} \qquad (1)$$

vector pairs or 84 vectors (L=3, K=9) and 91 vectors (L=2, K=14) for the 2 cases considered. Standard error-correction encodings are not best in APs.

4.5 P'_C VS. P_C

If P_C were used as our performance measure rather than P'_C, we obtained M/N values 30-50% larger. With CAAP encoding (which is necessary for performance exceeding that of the DSNN), P_C is meaningless.

4.6 QUANTIZATION TESTS

To demonstrate that analog accuracy suffices, we quantized the input, memory, and output elements to B bits. For the results in this section, we used unipolar (0,1) analog keys. To keep the input SNR at the same levels as in our other tests, we halved the input noise levels to $\sigma_n = 0.025$, 0.05 and 0.10. We found no performance loss with 6, 8, 8 (input, memory and output) bits for $\sigma_n = 0.025$, 0.05 and 0.10. For 6, 6, 8 bits, we find no performance loss for $\sigma_n = 0.05$ and 0.10 (when the HK-2 CAAP with binary-encoding was used with $\sigma_{syn} = 0.05$). Thus, we conclude that 6-8 bit accuracy suffices and that σ_{syn} can control this. We also verified the $\sigma_n = \delta/12^{1/2}$ relationship between σ_n and the number of levels present in the processor.

4.7 NUMBER OF HK ITERATIONS

We now consider our new HK algorithm that significantly reduces the number of HK iterations required (these iterations are only required in synthesis, classification is a one-pass system) and quantitative data on how M/N affects whether the HK-2 algorithm improves the pseudoinverse-2 solution (if it yields $P'_C = 100\%$ on noiseless data, no HK iterations occur). We consider only the HK-2 CAAP with N = 50 random analog keys. In the original HK algorithm [4], we used $\rho = 0.5$ in synthesis and we iterated on the matrix as a whole. Since each row of $\underline{M}$ converges at a different rate, we now iterate on each row separately and use $\rho = 0.95$. Table 8 shows the average number of HK iterations as a function of M/N for the five different output encodings. Each entry is the average of 10 memory matrices and are averaged over each memory row for each of the 10 matrices. Entries less than 1 indicate that an occasional HK iteration occurred, but generally the HK algorithm was not used (this occurs for M/N < 1.12). For M/N > 1.12, entries above 1 occur indicating that HK iterations improve the pseudoinverse-2 solutions. These numbers are all very small (a factor of 100 less than in the original algorithm [4]).

4.8 σ_{syn} SELECTION

In tests, we showed that the noise performance is optimum at the σ_{syn} used, that good noise performance results for lower σ_n, and that one should use σ_{syn} no larger than is necessary.

4.9 BINARY VS. ANALOG KEYS

To compare M/N when binary vs. analog keys are used, we synthesized an HK-2 CAAP with L=2, K=14 encoding with N = 50 bipolar binary keys with $\sigma_{syn} = 0.4$. The $\sigma_n = 0$ analog and 0-bits flipped data in Table 9 show that storage is larger when analog keys are used. This occurs because we used $\sigma_n = 0.4$ for the binary keys vs. $\sigma_n = 0.1$ for the analog keys. Tests with 0 to 3 input bits flipped show that our robust algorithms are also useful with binary keys ($\sigma_{syn} = 0.4$ corresponds to an $SNR_I = 8$dB or 2 of the 50 bits in error). As seen, L-max encoding yields preferable results to binary encoded recollections.

5. SUMMARY AND CONCLUSIONS

We have advanced several issues we feel are important in APs. The need to store $M > N$ key/recollection vector pairs and handle linearly dependent keys, and the need for analog feature space key vectors are among the most important. We gave two new performance measures: storage density (not just M) and P_C' (not P_C). We find that no 1:1 AP can compete with the DSNN unless alternative recollection vector encoding schemes are used. In such cases, CAAPs result and P_C' must be used (as P_C is meaningless). We showed that CAAPs offer an order of magnitude improvement in storage density, that robust HK-2 APs are preferable, and that their σ_{syn} parameter should not be selected larger than is necessary.

Various other point advanced are now highlighted. When $M < 1.12N$, we find that the HK iterations do not significantly improve the performance of the pseudoinverse AP. For CAAPs, we find the HK improvement to be more noticeable. For many:1 APs, this improvement will be even larger. Our revised HK algorithm requires much fewer iterations than the original algorithm. We find that all APs allow quantization with 6-8 bit inputs and memories being sufficient and that the robust algorithms can control the accuracy required. We find L-max and binary encoding to be preferable since no post-processing is required (L-max encoding yields better performance because of the sparse vectors it produces). Standard error-correcting techniques do not appear to be best for APs. If the CAAP is used to access a standard memory, then a large capacity error correcting general memory results (suitable for large knowledgebase searches).

ACKNOWLE!)GEMENT

The support of this research by a grant from the Air Force Office of Scientific Research is gratefully appreciated.

REFERENCES

1. B. Telfer and D. Casasent, "Ho-Kashyap Content-Addressable Associative Processors", IJCNN'90 (*International Joint Conference on Neural Networks*), IEEE Catalog No. 90CH2879-5, June 1990, San Diego, Vol. I, pp. I-751 - I-756.
2. B. Telfer and D. Casasent, "Ho-Kashyap Advanced Pattern Recognition HeteroAssociative Processors", Proc. SPIE, Vol. 1347, July 1990.
3. B. Telfer and D. Casasent, "A Closure Associative Processor", IJCNN International Joint Conference on Neural Networks [IEEE Catalog Number 89CCH2765-6], June 1989, Washington, D.C., Vol. 1, pp. I-99-103.
4. B. Telfer and D. Casasent, "Ho-Kashyap Optical Associative Processuīs", Applied Optics, Vol. 29, pp. 1191-1202, 10 March 1990.
5. K. Murakami and T. Aibara, "An Improvement on the Moore-Penrose Generalized Inverse Associative Memory", IEEE Trans. Soc. Man & Cybern., Vol. SMC-17, No. 4, pp. 699-707, July/August 1987.
6. K. Murakami and T. Aibara, "Least Squares Associative Memory and a Theoretical Comparison of its Performance", IEEE Trans. Soc. Man & Cybern., Vol. SMC-19, No. 5, pp. 1230- 1234, September/October 1989.
7. B. *Telfer and D. Casasent*, "Updating Optical Pseudoinverse Associative Memories", Applied Optics, Vol. 28, pp. 2518-2528, 1 July 1989.

8. S. Lin and D.J. Costello, Error Control Coding: Fundamentals and Applications, Prentice-Hall, Englewood Cliffs, NJ, 1983.
9. J. Austin and T. Stonham, "Distributed Associative Memory for Use in Scene Analysis", Image and Vision Computing, Vol. 5, No. 4, pp. 251-260, Nov. 1987.

MEMORY TYPE	STORAGE CAPACITY M	MEMORY SIZE NK	STORAGE DENSITY M/NK	STORAGE DENSITY (N=50)
DSNN	$M \rightarrow \infty$	NM	1/N	0.020
Correlation	$N/4\ln N$	N^2	$1/4N\ln N$	0.001
Pseudoinverse	N	N^2	1/N	0.020
HK	2N	N^2	2/N	0.040
Pseudoinverse CAAP	N	$N\log_2 N$	$1/\log_2 N$	0.177
HK CAAP	2N	$N+N\log_2 N$	$2/(1+\log_2 N)$	0.301

TABLE 2: Theoretical storage capacities and densities for different 1:1 associative processor models for exact key inputs.

σ_n	DSNN	CORREL	PSEUDOINV	HK	PSEUDOINV-1	HK-1	PSEUDOINV-2	HK-2
0.00	1,2	0.12	1.04	1.52	0.84	1.52	0.96	1.12
0.05	1,2	0.12	0.88	0.88	0.84	0.88	0.92	0.92
0.10	1,2	0.12	0.72	0.72	0.76	0.76	0.80	0.80
0.20	1,2	0.12	<0.6	<0.6	<0.6	<0.6	<0.6	<0.6

TABLE 3: Maximum storage M/N for $P_C' > 95\%$ for eight APs (with N = K = 50). (The DSNN used M = K = N and M = K = 2N. For the robust APs, $\sigma_{syn} = 0.1$).

σ_n	DSNN	CORREL	PSEUDOINV	HK	PSEUDOINV-1	HK-1	PSEUDOINV-2	HK-2
0.00	0.02	0.0024	0.021	0.030	0.017	0.030	0.019	0.022
0.05	0.02	0.0024	0.018	0.018	0.017	0.018	0.018	0.018
0.10	0.02	0.0024	0.014	0.014	0.015	0.015	0.016	0.016
0.20	0.02	0.0024	<0.012	<0.012	<0.012	<0.012	<0.012	<0.012

TABLE 4: Storage densities M/KN for $P_C' > 95\%$ for the eight APs in Table 3 (with K = N = 50).

σ_n	DSNN	CORREL	PSEUDOINV	HK	PSEUDOINV-2	HK-2
0.00	1,2	0.36	1.20	>1.60	1.12	>1.60
0.05	1,2	0.32	0.92	0.92	1.08	1.32
0.10	1,2	0.24	0.80	0.84	0.92	0.92
0.20	1,2	0.20	<0.60	<0.60	<0.60	<0.60

TABLE 5: Maximum storage M/N for $P_C' > 95\%$ for six CAAPs with N = 50 random analog inputs and K = 7 binary (± 1) encoded recollections. The DSNN used M = K = N and M = K = 2N. The robust APs used $\sigma_{syn} = 0.1$.

σ_n	DSNN	Correl	Pseudoinv	HK	Pseudoinv=2	HK-2
0.00	0.02	0.051	0.171	>0.229	0.160	>0.229
0.05	0.02	0.046	0.131	0.131	0.154	0.189
0.10	0.02	0.034	0.114	0.120	0.131	0.131
0.20	0.02	0.029	<0.086	<0.086	<0.086	<0.028

Table 6: Storage densities M/KN for $P'_C > 95\%$ for the six CAAPs in Table 5.

P'_c	σ_n	Recollection Vector Encoding (7 Data Bits)				
(%)		Binary	Hamming	BCH	L-max (L ones)	
		$(K = 7)$	$(K = 11)$	$(K = 15)$	$K = 14 (L = 2)$	$K = 9 (L = 3)$
	0.00	1.84	1.48	1.56	1.68	1.64
>95	0.05	1.36	1.40	1.52	1.64	1.48
	0.10	0.92	1.20	1.36	1.48	1.20
	0.20	0.60	0.76	0.84	0.96	0.72

Table 7: 1:1 CAAP tests of different recollection vector encodings. The maximum M/N for $P'_C > 95\%$ is given for the HK-2 CAAP with N = 50 random analog keys and $\sigma_{syn} = 0.1$.

$M/N = M/50$	0.80	0.88	0.96	1.04	1.12	1.20	1.28	1.36	1.44	1.52	1.60
binary encoded (K=7)	0	0.0143	0.0186	0.124	0.339	1.09	1.05	2.29	3.74	5.56	8.08
L-max(L=2,K=14)	0	0.0229	0.154	0.504	2.20	5.39	9.93	15.7	22.2	25.6	28.1
L-max(L=3,K=9)	0	0.0611	0.0756	0.627	1.38	3.61	6.97	10.4	15.2	19.7	23.8
Hamming (K=11)	0	0.0227	0.0491	0.475	1.72	5.03	9.61	14.7	21.2	26.6	28.2
BCH (K=15)	0	0.0140	0.0560	0.620	1.95	4.51	9.55	15.9	21.4	27.0	28.2

Table 8: Average number of iterations required to synthesize an HK-2 CAAP for different M/N values and different recollection vector encodings with N=50 analog keys.

		BINARY ENCODED OUTPUTS				L=max(L=2) ENCODED OUTPUTS			
	Analog Keys	BINARY BIPOLAR KEYS (number of errors)				BINARY BIPOLAR KEYS (number of errors)			
P'_c	$\sigma_n = 0.0$	0	1	2	3	0	1	2	3
>90%	2.04	1.52	0.80	0.60	<0.6	1.68	1.36	1.04	0.92
>95%	1.84	1.32	0.68	<0.6	<0.6	1.44	1.16	0.96	0.72
>99%	1.52	0.92	0.60	<0.6	<0.6	1.24	1.04	0.72	<0.6

Table 9: Maximum M/N for which the given P'_C is obtained for N = 50 binary bipolar keys (with 0 to 3 input bit errors) with two different recollection vector encodings for an HK-2 CAAP with $\sigma_{syn} = 0.4$ (for comparison, analog key data with $\sigma_{syn} = 0.1$ are included).

CHAPTER 3

"Ho-Kashyap Advanced Pattern Recognition HeteroAssociative Processors"

Proc. SPIE
Volume 1347
San Diego
July 1990

Ho-Kashyap Advanced Pattern Recognition Heteroassociative Processors

David Casasent and Brian Telfer
Center for Excellence in Optical Data Processing
Department of Electrical and Computer Engineering
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

We review different categories of associative processors with attention to the properties of their key and recollection vectors, the test procedures to be used and the performance measures to be used to compare various associative processors. We review new pseudoinverse and Ho-Kashyap associative processors and robust versions of each. Quantitative data is presented on the performance of these new pattern recognition associative processors. In all cases we show significant improvement over prior data with $M \gg N$ (M is the number of key/recollection vectors pairs stored and N is the dimensionality of the input key vector). Quantization of the number of analog levels and comparisons of various recollection vector encodings are considered.

1 Introduction

We consider optical associative processors (APs) for pattern recognition (PR). The basic architecture used (Figure 1) is the standard optical matrix-vector processor. P_1 contains the input key vector (dimension N), the associative processor matrix of size $K \times N$ is at P_2, and the output recollection vector (of dimension K) is at P_3. The number of key/recollection vector pairs stored is M. Section 2 advances vital remarks on the various types of APs, key and recollection vector properties of each, test procedures for each and performance measures to be used for comparing various APs, key and recollection vector properties of each, test procedures for each and performance measures to be used for comparing various APs. Section 3 briefly reviews the various AP synthesis algorithms we consider and Section 4 provides quantitative test data.

2 Associative Processors (Performance Measures, Test Procedures and Comparison Issues)

The major types of APs are noted and remarks are advanced on the nature of the input/output key/recollection vectors for each, the test procedures to be used with each (Sections 2.1-2.7), and performance measures and comparison measures (Section 2.8) to be used.

2.1 General Remarks

We note that analog input data are essential for optical APs to compete and that AP algorithms must inherently include (within the algorithm) techniques to allow limited analog accuracy. Our robust and error-correcting algorithms achieve this. All APs we consider use one pass (not iterative recall) and are preferable. We first advance remarks on APs for image data. No AP (except the symbolic correlator neural network (NN) production system [1,2]) allows use of multiple input objects in the field of view in parallel. APs and NNs using iconic (pixel-based) input neuron representations are not attractive and feature space input neurons should be used (to reduce neuron dimensionality and storage M requirements).

All AP matrices M are analog. We test all APs for analog performance using quantized M values (and quantized input and output values). Quantized output values allow output WTA (winner take all) or maximum selection post processors to have output WTA errors. These quantization tests test the APs' analog (optical) performance. For optical implementations, negative values must be encoded. All AP matrices are bipolar. This requires twice the number of key and recollection elements (or twice the processing time plus the need to alter M for positive M_+ and negative M_- values). Thus, we use space multiplexing of M data in all cases. Bias and other [3] methods increase M dynamic range requirements, but may [3] be useful to overcome certain optical component bias errors.

The amount of noise and the nature of the noise added to the key inputs varies with the application. For input P_1 SLMs (spatial light modulators) or point modulators, the noise standard deviation σ_n is related to the separation δ between useful gray levels by $\sigma_n = \delta/12^{\frac{1}{2}}$. For SLMs or point modulators with 2, 4, and 6 bits of gray scale and a full scale value of 1, we find $\sigma_n = 0.072$, 0.018, and 0.0045. For a given analog accuracy $MPFSE$ (maximum percent full scale error), we find $\sigma_n = (MPFSE)(FS)/(3 \times 100\%)$, where FS is the full scale value (typically 1). This is the 3σ 99% confidence value. Thus, for a 1% analog processor which is typical ($MPFSE = 1\%$), we find $\sigma_n = 0.003$ or $\sigma_n = 0.003$ (with $\delta = 0.01$) will model this. All of these σ_n accuracy values are quite low and thus input noise σ_i added to the key vectors will be larger (we use $\sigma_i = 0.1$) and due to noise in the input image data (not processor errors). Our robust algorithms include σ_i directly in the algorithm and calculate M for best performance in that σ_i noise. For simplicity, we add noise to the input image feature vectors (rather than the images). We use uncorrelated noise (UCN) of zero-mean with negative noise pixels set to zero. We renormalize the input feature vector after noise is added (this is required for distortion invariance (DI)). White Gaussian noise (WGN) and UCN are realistic for the point noise in characters. Dirt, clouds, image clutter, etc. are modeled by correlated noise (CN) and blobs (false class and discrimination tests are somewhat suitable here).

The ability to easily update (add and delete key/recollection vector pairs) is an issue but is not a major real time concern. We have addressed this [4] and note that our new robust-2 algorithm easily allows this and that in multiple AP cases, only one AP need be updated.

Key vector requirements are a major issue that we consider (and that most researchers ignore). For exact solutions, standard correlation matrix AAPs require orthonormal keys and pseudoinverse/MSE (minimum squared error) APs require linearly independent keys. Both assumptions are unrealistic. We consider linearly dependent keys (in our Ho-Kashyap (HK) APs [5,6]).

In this paper, we consider only linear decision surfaces. Our minimum-cost APs (MC APs) allow nonlinear decision surfaces. These are essential for all APs (especially for PR cases where class separation is the important issue).

We now advance several new classifications of APs and discuss each briefly.

2.2 PR HAPs – Small Problems

For pattern recognition (PR), heteroassociative processors (HAPs) are used. Their outputs denote the class of the input object. The input key vectors should be analog (feature space) and the output recollection vectors should be unit or binary-encoded vectors. For small problems, unit vectors are preferable (if WTA detection is used) but binary vectors are adequate (they require a fixed detection threshold T). Binary encoding results in a smaller AP matrix than do unit vectors (both are very similar in size for small problems with small K). With small K, WTA postprocessing is acceptable (and trivial). For 1:1 APs as in Section 2.4 (each key vector is associated with a separate recollection vector), we have shown that unit vectors are preferable [7]. Small problems refer to the size K of the recollection vector (the number of classes) and to the types of distortions required (full 3-D distortions require very large M).

2.3 PR HAPs – Large Distortion Problems

When a wide range of 3-D object distortions must be handled for a $C = 3$ class problem, we find (Section 4.3) that neither binary nor unit vector encoding is acceptable and that hyperplane-selected recollection vector encoding (a minimum cost, MC, AP) is required. In the PR cases in Sections 4.2 and 4.3, many key vectors (distortions in one class) are assigned to the same recollection vector.

For all distortion-invariant PR problems (small and large), we test these APs using noisy training images and using intermediate distorted images (at intermediate distortions between training images, i.e. with data not present in the training set).

2.4 PR HAP – Large Class (1:1 General HAP)

In large class problems (C large) when limited distortions (e.g. in plane) are required, such that only one feature vector per class is required, then each key vector (one per class) is assigned to a different recollection vector. We refer to this as a 1:1 General AP. Since generating data for this case is generally prohibitive, we use analog keys uniformly distributed [0,1] or [-1,+1] and random (binary encoded) recollection vectors with elements ± 1 or 0,1. We test these HAPs with noise present in the input keys. Modifications in which several keys map to the same recollection vector are possible. Section 4.6 includes tests of various recollection vector encodings besides binary.

2.5 Autoassociative Processors (AAPs)

This class of AP has been the most widely analyzed. They require $\mathbf{Y} = \mathbf{X}$ and are useful for producing full outputs ($\mathbf{y}$) from partial/noisy inputs ($\mathbf{x}$). For binary cases, $\mathbf{x}$ and $\mathbf{y}$ are uniformly distributed binary [0,1] and for analog cases $\mathbf{x}$ and $\mathbf{y}$ are uniformly distributed analog [0,1]. They are tested with input noise and partial inputs. We do not consider such APs and do not find a use for AAPs (as we now discuss). For image processing, analog $\mathbf{x}$ and $\mathbf{y}$ are used. But the reason for image processing is PR and thus we use a HAP and can test it on noisy/partial inputs. A cascade of an AAP and HAP does not appear to be better than an HAP alone (a nonlinear (NL) function is not easily provided in an analog AAP and thus the linear AAP function can be included in the HAP). Of more concern is the large size required for any AAP (with image pixels) which makes it prohibitive and correlation or morphological processing [8] preferable especially if all image distortions are required to be included in the AAP (here M is excessive when iconic (pixel-based) input P_1 neurons are used). Similar remarks apply to binary AAPs for image processing (with the additional note that most imagery, except text, is not binary, but with font and point size variations M is again excessive). An extra NL function is possible with binary P_1 data and the cascade of such an AAP/HAP may be preferable (but the AAP size and its M appear excessive). For general binary storage our label/standard memory AP [6] is preferable (Section 2.6). For general data, the conventional digital encoding does not allow for use of the large error correction possible with APs (vs. standard digital error correction techniques) since all digit combinations of data are possible. For knowledge bases, our label/standard memory appears preferable (Section 2.6). For these reasons, we find no advantages and uses for AAPs.

2.6 Label/Standard Memory AP

Figure 2 shows this AP concept in which a HAP outputs a label that denotes the location in a standard addressable memory of the associated input data. A standard memory cannot tolerate errors while the HAP input processor can. We consider this AP with analog uniformly distributed [-1,1] or [0,1] keys and random (binary) (± 1) recollection elements as in Section 2.4. This is useful for knowledge bases, attributes (with 1:1 modifications) etc. and possibly in speech and spelling checker AP cases. The standard memory outputs can be feature/attribute/association lists, etc. associated with the input object. The standard memory data can be stored analog or digitally.

2.7 Multiple APs

Many problems will require use of multiple APs and a cluster [9] etc. front end. Our present concern is the individual APs.

2.8 Performance Measures

The performance measures we use differ vary significantly from most prior ones (for reasons indicated). The three major performance measures we use follow.

2.8.1 Storage Density

The number of key/recollection pairs M stored is the major concern. M must be large but the memory size KN required is also of concern. Thus a storage density M/KN with large M is the key parameter. Many researchers consider only M and employ higher-order etc. APs or neural nets (NNs) to increase M with an associated increase in KN (thus achieving no improvement in storage density). The majority of prior APs store $M \ll N$ and are thus not competitive. Our 1:1 General APs store $M > N$ ($M = 2N$ is the theoretical maximum). Our PR APs store $M \gg N$.

2.8.2 Radius of Attraction

Various measures such as σ_o^2/σ_i^2 (output-to-input noise variance) are not suitable for HAPs [7]. Similarly, Hamming distance (and the related standard radius of attraction) is useful only for binary keys. With analog keys, we use Euclidean distance as radius of attraction (this is the largest distance that an input can be from a key vector and still be reliably identified as that key). We use this measure only for our 1:1 General HAP (where noise σ_i sets the variations and where each key is a separate point in feature space). See Section 2.8.3. Our radius of attraction thus relates to σ_i. It is specified for a given P_c (see Section 2.8.3).

2.8.3 Recall Accuracy

In complex PR cases (with many 3-D distortions per object) the probability distributions per class vary widely and are not spheres. Thus, radius of attraction is not meaningful nor is it defined. The probability distributions determine the performance expected and the recollection vector encoding used. Thus, for all APs, we use recall accuracy as our major performance measure. Few researchers use this and when it is used, they use P_c (the percent of recollection vector elements correctly recalled). This is useful only for AAPs but is useless for HAPs. We use P_c' (the percent of all recollection vectors with perfect recall) which is more meaningful (and less than P_c).

3 AP Review

We now highlight the various APs we consider. This review concerns the synthesis algorithms used to calculate the AP matrix M, key vector requirements, noise and analog accuracy considerations. Hopfield [10,11] and Direct Storage Nearest Neighbor (DSNN) [12] APs are detailed elsewhere. AAP and HAP storage capacity [7], Pseudoinverse APs, and Ho-Kashyap (HK) PR APs [5] and HK 1:1 General HAPs [6] are detailed elsewhere. BAMs are not considered as they have the limited $M \ll N$ storage of the Hopfield memory [5]. All APs solve $\mathbf{M}\mathbf{x}_m = \mathbf{y}_m$ for M for all $(\mathbf{x}_m, \mathbf{y}_m)$ pairs. The minimum squared error (MSE) solution is $\mathbf{M} = \mathbf{Y}\mathbf{X}^+$ (where the columns of the matrices X and Y are the $\mathbf{x}_m$ and $\mathbf{y}_m$ and $\mathbf{X}^+ = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T$ is the pseudoinverse).

The correlation HAP uses $\mathbf{M} = \mathbf{Y}\mathbf{X}^T$ which is not realistic as $\mathbf{X}^T = \mathbf{X}^+$ only if the keys are orthogonal. We include it as it is one of the most used APs although its capacity and performance (it is not a nearest neighbor AP) are not attractive.

The DSNN HAP uses $\mathbf{M} = \mathbf{X}^T$ and is the only nearest-neighbor HAP. It has the best performance of any HAP, but its disadvantage is its large size and it requires a WTA (which is not realistic for the large K it needs).

The standard pseudoinverse AP uses $\mathbf{M} = \mathbf{Y}\mathbf{X}^+$. This minimizes the error $J_1 = \| \mathbf{Y} - \mathbf{M}\mathbf{X} \|^2$ with respect to M and has useful performance with much larger storage ($M \approx N$) than the correlation HAP. However, the Y solutions are only exact for linearly independent keys (which is not realistic). The MSE pseudoinverse AP is the same and is used when $M > N$. Here the keys are linearly dependent and the solution is the MSE one. This still allows useful solutions when $M < N$ and when $M > N$ (and hence much larger storage) but it has poor noise performance when $M \approx N$. We have used two robust solutions for improved performance when noise is present. The Robust-1 Pseudoinverse HAP (we refer to this as a Pseudoinverse-1 AP) uses $\mathbf{M} = \mathbf{Y}\tilde{\mathbf{X}}^+$ where $\tilde{\mathbf{X}}$ has small singular values with $\mu < \sqrt{M}\sigma_i$ set to zero. This improves noise performance for all M/N and especially for $M \approx N$. It thus allows useful solutions in noise for all M including $M > N$. The Robust-2 Pseudoinverse HAP (we refer to this as a Pseudoinverse-2 AP) yields the optimal solution when input noise (uncorrelated) is present with a standard deviation σ. The solution is $\mathbf{M} = \mathbf{Y}\mathbf{X}^T(\mathbf{X}\mathbf{X}^T + M\sigma^2\mathbf{I})^{-1}$, where the noise correlation matrix $\sigma^2\mathbf{I}$ is included. This minimizes $J_2 = \| \mathbf{Y} - \mathbf{M}(\mathbf{X} + \mathbf{N}) \|^2$ with respect to M. This provides better storage capacity and the optimum input noise performance.

All prior APs do not work well with linearly dependent keys (hence they are typically limited to $M < N$ for high recall accuracy). The HK AP [5] allows excellent performance and storage ($M > N$) with linearly dependent keys. For general keys (our 1:1 General HAP), it stores $M = 2N$ (the maximum for any 1:1 General HAP) [13]. The HK AP algorithm minimizes $J_1 = \| \mathbf{Y} - \mathbf{M}\mathbf{X} \|^2$ with respect to both M and Y. It thus changes the Y elements to optimize J_1. This is done only in synthesis. In recall, the same binary recollections are used (after thresholding). The five algorithm steps are shown in Table 1. The algorithm starts (Step 1) with the MSE pseudoinverse solution and refines (improves) it (producing a better MSE for the case of linearly dependent keys). The error matrix $\mathbf{E}_n$ in Step 2 is the error in the $\mathbf{y}_m$ (the $\mathbf{y}_m$ obtained are not exact since the solution is the MSE one). The modified error matrix $\mathbf{E}'_n$ (Step 3) equals $\mathbf{E}_n$ except that all $\mathbf{E}_n$ elements that differ in sign from the corresponding elements in $\mathbf{Y}_n$ are set to zero (the S matrix contains the signs of the desired outputs in Y and $\otimes$ denotes Hadamard multiplication). The initial $\mathbf{y}_m$ choices are then modified (Step 4) and the algorithm iterates until $\mathbf{E}'_n = \mathbf{0}$. Convergence has been proven [5] for linearly separable data (where $\mathbf{E}_n = \mathbf{0}$) and for linearly nonseparable data (where $\mathbf{E}'_n = \mathbf{0}$). This HK AP algorithm offers a solution for $M > N$ that is better than the MSE pseudoinverse solution (for linearly dependent keys).

Step	Operation
1	$\mathbf{M}_n = \mathbf{Y}_n\mathbf{X}^+$
2	$\mathbf{E}_n = \mathbf{M}_n\mathbf{X} - \mathbf{Y}_n$
3	$\mathbf{E}'_n = \frac{1}{2}(\mathbf{E}_n + \mathbf{S} \otimes \lvert\mathbf{E}_n\rvert)$
4	$\mathbf{Y}_{n+1} = \mathbf{Y}_n + 2\rho\mathbf{E}'_n, \quad 0 < \rho < 1$
5	If $\mathbf{E}'_n \neq \mathbf{0}$ go to 1.

Table 1: Ho-Kashyap AP Algorithm

We have developed two robust (good input noise performance) versions of the standard HK algorithm. The robust-1 HK HAP starts (Step 1) with $\tilde{\mathbf{X}}^+$ (the pseudoinverse of $\tilde{\mathbf{X}}$). The robust-2 HK AP starts with the robust-2 pseudoinverse solution for $\mathbf{X}^+$ in Step 1. The final modification that we have deve'oped is the error-correcting HK AP [5]. In this case we update Y using $\mathbf{Y}_{n+1} = \mathbf{Y}_n + \mathbf{E}'_n$ and we update M using $\mathbf{M}_{n+1} = \mathbf{M}_n + \rho(\mathbf{S} \otimes \lvert\mathbf{E}_n\rvert)\mathbf{X}^T$. This allows calculation of M on a low accuracy (optical) processor and produces an M that should have low accuracy requirements.

With a better MSE, one cannot show that P_c or P_c' will be better, but we have observed this to occur (when noise is present). We note that the robust algorithms can give slightly lower performance with no noise, but that they give better performance in noise.

4 Pattern Recognition HAP Test Results

We now present various PR HAP test results.

4.1 Feature Space Neurons (In-Plane DI)

To reduce neuron dimensionality (N) and storage (M), we use feature space key neuron representations. The feature space we consider in this paper is a wedge-ring detector (WRD) sampled $|\mathrm{FT}|^2$ (Fourier transform = FT) space [14] (see Figure 3). These neurons are shift invariant (SI) (since the $|\mathrm{FT}|^2$ is), the wedge neurons are also scale invariant and the ring neurons are also rotation invariant (RI). SI is essential to avoid the need to include all shifted objects in the training set M. For RI, we use the ring samples and only one training image per class. For scale invariance, we use the wedge samples and only one training image per class. For complete in-plane invariance, we use adjunct techniques (moments or a priori information) to locate the nose of an aircraft or the orientation of an object and thus employ only wedge $|\mathrm{FT}|^2$ samples. Figure 4 shows test results of a $N = 32$ element key (32 wedge samples), $K \times N = 18 \times 32$ matrix, $K = 18$ element recollection vector HAP in which outputs in elements 1-9 denote Phantom jets and outputs in elements 10-18 denote DC-10 aircraft (the output neuron element activated gives the object's rotation). The system was trained on 9 images per class (20^o rotations), covering 180^o (by symmetry the other 180^o images have the same feature space). The outputs are shift and scale invariant and AP synthesis requires two training images (one per class) since the 9 rotations of each object (at 20^o intervals) are simply cyclic shifts of the 1-D wedge $|\mathrm{FT}|^2$ inputs at 0^o.

4.2 3-D DI Test Sets 1-3

We now consider three test cases involving three classes of aircraft and various amounts of pitch and roll distortions. Figure 5 shows several of the distorted images per class with the top-down 0^o pitch and 0^o roll view in the center top and with pitch increasing to the left (at -40^o and -80^o) and right ($+40^o$ and $+80^o$) and with roll increasing down (40^o and 80^o) from the central images. In these data, a 33 element key vector was used with one element being a constant 1 (this allows the threshold to be adjusted, i.e. the associated hyperplane to be shifted) and with 32 wedge $|\mathrm{FT}|^2$ samples. Thus $N = 33$. The inputs are 128×128 and are not edge enhanced. The pixels in the center of the FT within a radius of 20 are set to 0 (to enhance higher frequencies). We use recollection vectors with $K = 2$ (Test Cases 1 and 2) and $K = 3$ (Test Case 3). The HAP matrices are thus 2×33 or 3×33. We use unit recollection vectors and WTA maximum selection (a "1" output in element 1 indicates a Phantom, a "1" output in element 2 indicates a DC-10 and a "1" output in element 3 (Test Case 3) indicates an F104). The three test cases involve different numbers of classes and different amounts of pitch and roll distortions: Test Case 1 ($\pm 50^o$ of pitch and roll, two classes), Test Case 2 ($\pm 60^o$ of pitch and roll, two classes) and Test Case 3 ($\pm 80^o$ of pitch and roll, and all three classes). Table 2 summarizes these three test cases. The training set size $M = N_T$ used to synthesize each HAP is noted (we use 441, 625, and 1089 images per class). These are all images in the pitch and roll range noted at 5^o intervals. The test set size used is also noted (400, 576, and 1024 images per class). These are images at 2.5^o rotation intervals between the training set image views. Representative wedge $|\mathrm{FT}|^2$ data are shown in Figure 6.

In all tests, the sum of the squares of the wedge data was normalized to 1 (after removing the low frequency 20-pixel radius). When noise was added to the wedge data, the standard deviation was $\sigma_i = 0.1$, negative valued pixels were set to zero, and the noisy wedge data (after removing pixels in the low frequency radius of 20) was renormalized. The new noise had $\sigma_i = 0.078$. Normalization is done in this way to provide invariant features

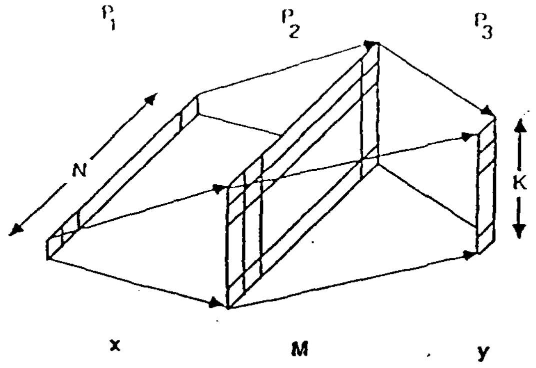


Figure 1: Basic optical matrix-vector associative processor

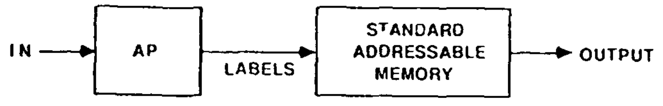
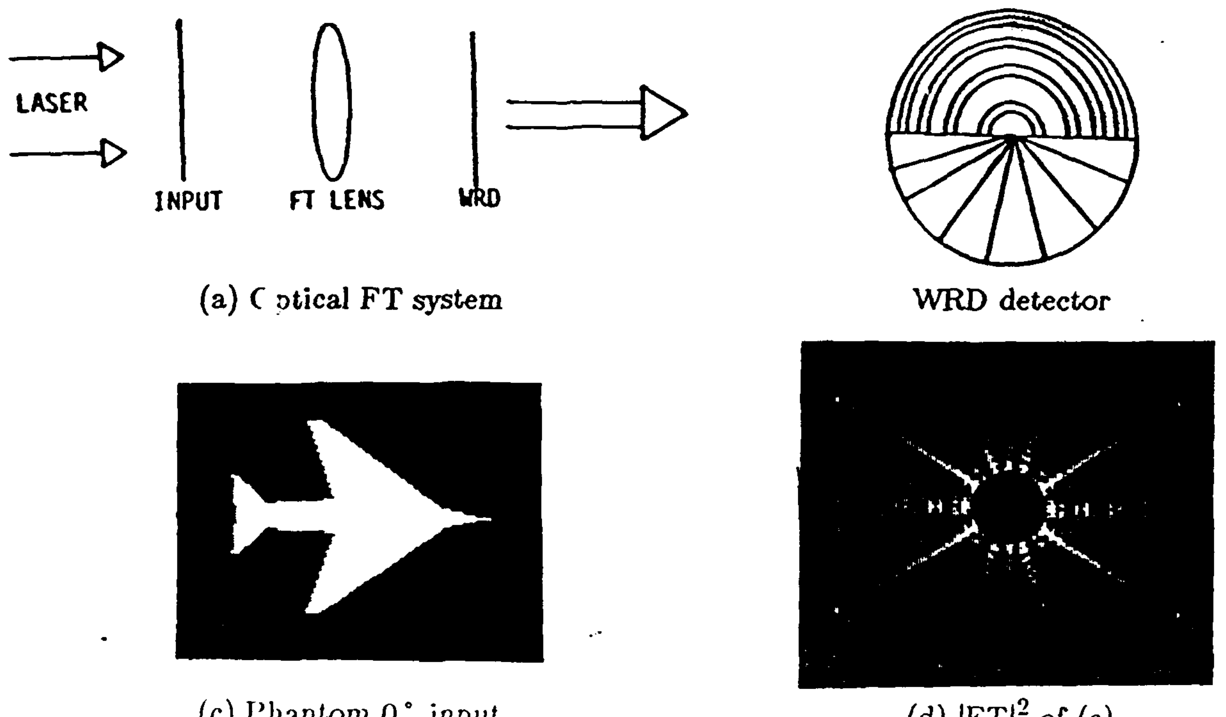


Figure 2: Label/standard memory associative processor



(a) Optical FT system

WRD detector

(c) Phantom 0° input

(d) $|FT|^2$ of (c)

Figure 3: WRD $|FT|^2$ feature space HAP input neuron representation space

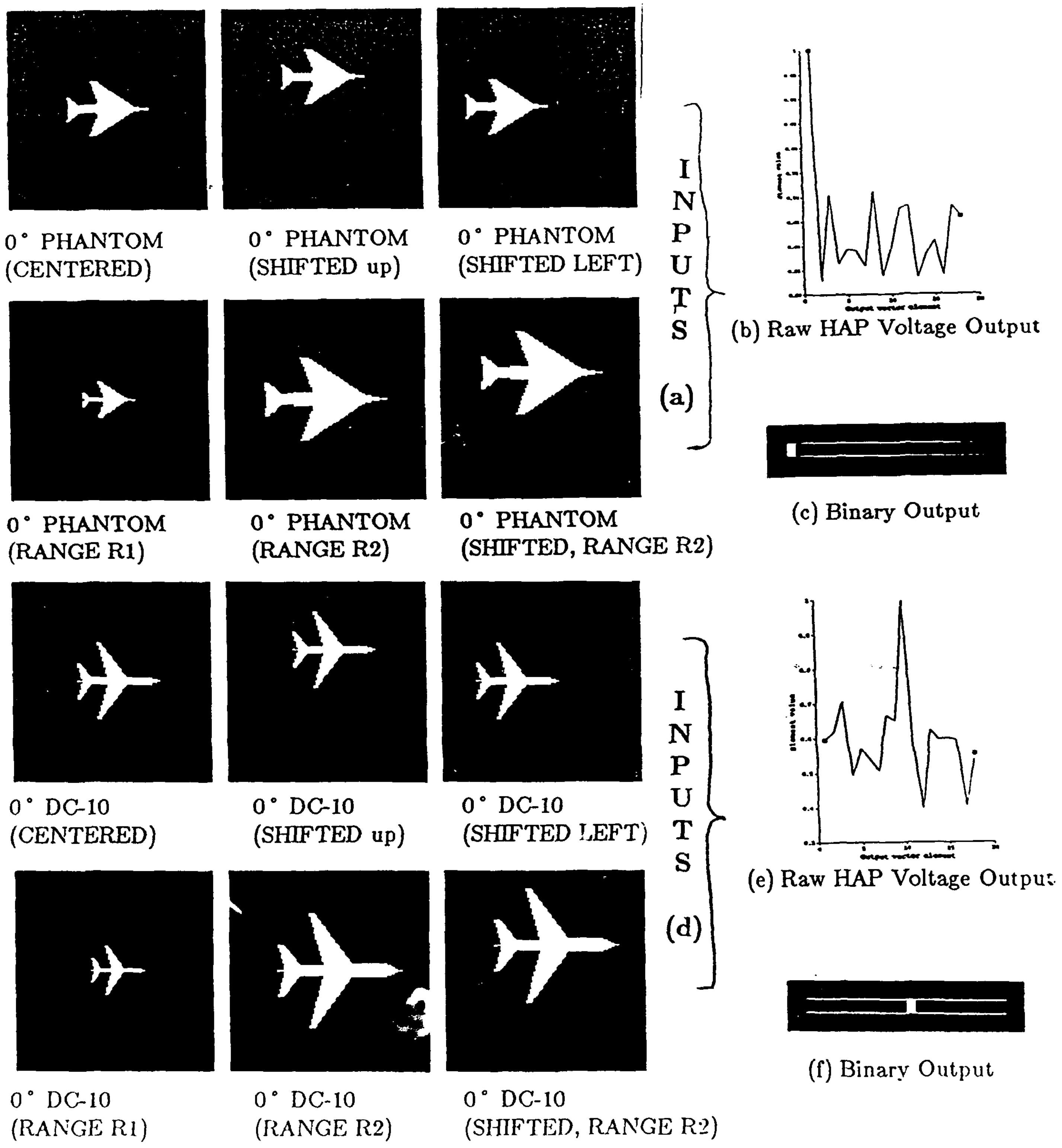
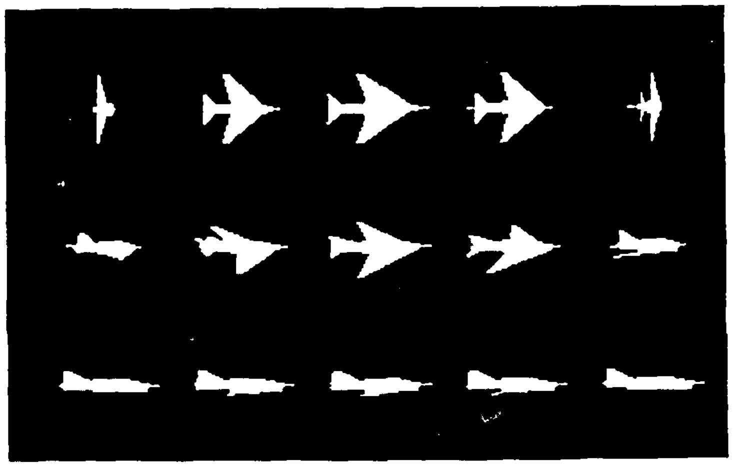
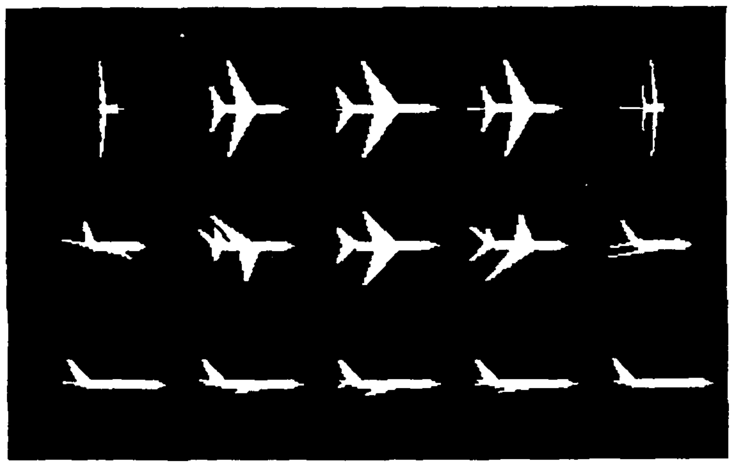


Figure 4: In-plane distortion-invariant wedge $|FT|^2$ HAP ($N = 32$ wedges, $K = 18$ element recollection)


(a) Phantom images


(b) DC-10 images

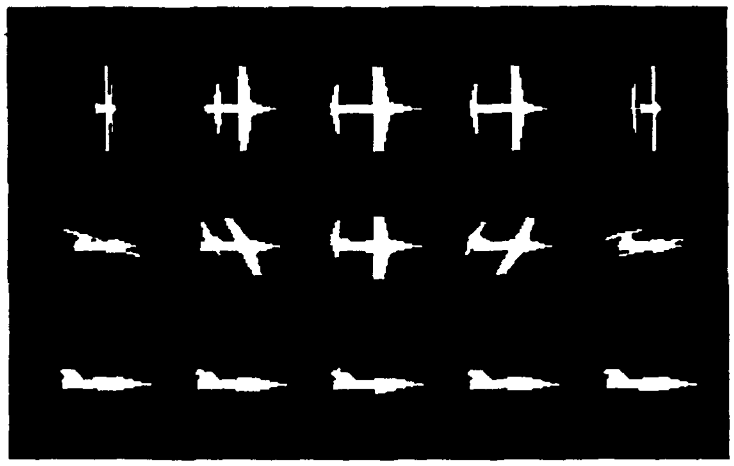

(c) F104 images

Figure 5: Representative roll (vertical $0^\circ, 40^\circ, 80^\circ$) and pitch (horizontal, $0^\circ, \pm 40^\circ, \pm 80^\circ$) 3-D distorted images of three aircraft

Test Set	Aircraft	Pitch and Roll Distortions	$M = N_T$ (Train Set) (5° increments)	Test Set (2.5° intermediate)
1	Phantom, DC-10	±50°	2 × 441 = 882	2 × 400 = 800
2	Phantom, DC-10	±60°	2 × 625 = 1250	2 × 576 = 1152
3	Phantom, DC-10, F104	±80°	3 × 1089 = 3267	3 × 1024 = 3072

Table 2: Three test sets for 3-D DI

(with object scale changes, etc.). The $\sigma_i = 0.078$ used corresponds to an SNR = 5 or 7.1 dB, where SNR = 10log(Average wedge energy/σ_i^2) = 10log[(1/32)/0.078^2].

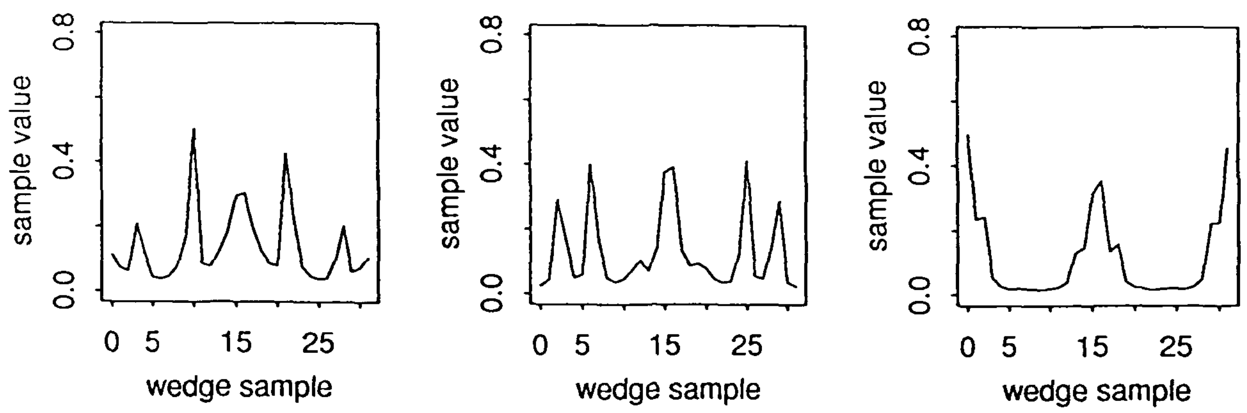
4.3 Initial Test Results (3-D Distortions)

Table 3 shows the results obtained. The correlation matrix performs worst as it requires orthogonal keys for $\mathbf{X}^{-1} = \mathbf{X}^T$ and an exact solution. This is the AP used in most HAP work. For this 3-D DI PR problem (many:1), the correlation AP performance is quite good (although its P_e is 25% larger than for advanced APs). This occurs because the correlation AP uses one LDF per class that is proportional to the average of the training set for one class and this performs well here (simply because the density distributions of the various distorted objects are approximately spherical and thus the VIP with the average vectors is correct (nearest neighbor) with WTA used). In our 1:1 general AP application, the correlation AP performance will be much worse than for our advanced APs. The DSNN is expected to perform best and it does, but its size is excessive since all training vectors are separate columns of the $\mathbf{M} = \mathbf{X}^T$ matrix. The size of all HAPs is 2 × 33 or 3 × 33 except for the DSNN. The size of the DSNN (where $\mathbf{M} = \mathbf{X}^T$) is 882 × 33, 1250 × 33 or 3267 × 33 (which are all excessive, and a WTA is needed).

We first consider the training and test set results from the Test Set-1 data with no noise. The pseudoinverse HAP yields 97.7% and 99% recall, but the HK gives perfect (100%) results on both the training and test sets. Only 14 HK iterations were required (these are off-line in synthesis and in real time only one pass is used). All results are excellent with test set results agreeing with those from the training set. The pseudoinverse is not exact and the HK improves it (with 14 iterations). The robust algorithms (Tests 4 and 5) are expected to provide less recall on non-noisy data. The data shows this as performance drops (98.1% and 99.5%) while the pseudoinverse-2 improves slightly to the same values. We expect the robust algorithms to be better in noise than in no noise. We see ($\sigma_n = 0.1$ data in Tests 4 and 5) better performance. The robust algorithm improves the noise performance of the pseudoinverse HAP from 93.1 to 94.3 (Tests 2 and 4) and of the HK AP from 93.2 to 95.1 (Tests 3 and 5) and the HK-2 HAP yields the best noise performance. Note that the improvements in the recall P_c' given are small but the percent error change is large (e.g. P_e in noise improves from 7.7% to 4.6% (tests 3 and 5) which is a 40% improvement).

The Test Set 2 results are similar with a lower P_c' indicative of a more difficult problem. The same general trends result. Here, the pseudoinverse (standard and robust) performs better than HK on the noisy test set. This may due to the fact that in this difficult PR problem, the distribution of the data determines performance moreso than the reduced MSE that HK gives (there is a larger drop in P_c' with noise here than in Test Set 1). The robust algorithms perform better than the standard ones in noise and the robust HK (HK-2) is best or close to best. The Test Set 3 results show P_c' too low to generally be of interest – thus no noise tests were performed on it and the robust algorithms were not attempted. For Test Sets 2 and 3, the data given is for 500 training iterations of the HK algorithm. The algorithm was terminated here. Comparable results occur after about 100 iterations.

Our advanced HAPs achieve excellent HK-2 results (P_e = 4.6-4.9% in Test Set 1 and P_e = 8.8-9.4% for Test Set 2) in high ($\sigma_n = 0.1$) noise with a very small 2 × 33 = 66 element memory. The memory size is much less than for the DSNN (882 × 33 = 29,106 and 1250 × 33 = 41,250 elements) and performance is much better than for the correlation HAP (P_e = 5.2-7.0% and 11.8-12.2% for Test Sets 1 and 2). These new memories stored M = 882



(a) Phantom (b) DC10 (c) F104

Figure 6: Representative wedge $|\mathrm{FT}|^2$ input neuron HAP data of aircraft at 0^o

Test	HAP	Test Set 1				Test Set 2				Test Set 3	
		Training Set		Test Set		Training Set		Test Set		Train	Test
		$\sigma =$ 0.0	$\sigma =$ 0.1	$\sigma =$ 0.0	$\sigma =$ 0.1	$\sigma =$ 0.0	$\sigma =$ 0.1	$\sigma =$ 0.0	$\sigma =$ 0.1	$\sigma =$ 0.0	$\sigma =$ 0.0
1	Correl	96.8	93.0	97.3	94.8	92.0	87.8	93.1	88.2	75.3	77.0
2	Pseudo	97.7	93.1	99.0	92.6	96.3	87.3	97.2	88.5	78.9	80.0
3	HK	100	93.2	100	92.3	99.4	82.3	99.3	85.5	80.7	81.2
4	Pseudo-2	98.1	94.3	99.5	95.0	95.7	90.2	96.4	91.7	-	-
5	HK-2	98.1	95.1	99.5	95.4	95.5	90.6	96.3	91.2	-	-
6	DSNN	100	99.5	100	99.3	100	97.9	100	98.4	100	98.8

Table 3: 3-D DI PR HAP Test Results (P_c')

HAP	No. Bits	Train	Test	Train	Test
		No Noise		$\sigma_n = 0.1$	$\sigma_n = 0.1$
	∞	99.4	99.3	82.3	85.5
HK	12	99.2	99.3	82.3	85.5
Assoc	10	99.4	99.3	82.3	85.5
Proc	8	99.3	99.3	82.4	85.6
	6	98.9	98.7	83.0	84.9
	4	93.3	93.8	75.5	76.8
	2	54.2	52.3	52.3	53.5
	∞	95.5	96.3	90.6	91.2
HK-2	12	95.5	96.3	90.6	91.1
Assoc	10	95.5	96.3	90.6	91.2
Proc	8	95.8	96.3	90.6	91.5
	6	95.8	96.3	91.5	91.9
	4	95.1	96.0	89.6	91.0
	2	86.6	87.3	78.6	78.8

Table 4: Quantization tests for typical HK APs (Test Set 2)

and $M = 1250$ training vectors with $N = 33$ ($M = 27N$ and $M = 38N$), which is much larger than for any prior HAP. They have been extensively tested on 800 and 1152 test images and with noise.

4.4 Quantization Tests (3-D Distortions)

The additive input noise tests (Table 3) are indicative of the low number of quantization levels required in the input data. We now quantized the input, matrix and output to the same number of levels. Output recollection vector quantization can cause errors with the WTA postprocessing performed. We found that all APs could be quantized down to 6 levels with negligible loss in P_c'. For some cases, quantization to 4 levels also gave an equally small drop in P_c'. For the Test Set 1 data, the pseudoinverse (Test 2) and HK (Test 3) APs could be quantized to 6 bits with no loss in P_c' for the no noise case and to 8 bits for the $\sigma_n = 0.1$ noise case. The HK AP (Test 3) could be quantized to similar levels. The robust APs (Tests 4 and 5) were slightly better allowing 4-6 bits of quantization with no loss. Table 4 shows specific data for the HK and HK-2 APs for Test Set 2 data. They show that 6 bits of data suffice.

4.5 1:1 General HAP (Large-class PR Tests)

For large-class problems with one feature vector sufficient per class (e.g. for in-plane distortions), we consider our 1:1 General HAP (Section 2.4) with each key vector associated with a separate recollection vector (hence we use the term 1:1 HAP) using analog [-1,1] uniformly distributed key vector elements and random (± 1) binary keys (hence we use the term General HAP).

4.5.1 Performance for $M \approx N$ in Noise

We consider an HAP with $N = K = 50$ (a $50 \times 50 = 2500$ element matrix). We increase M (the number of stored key/recollection vector pairs) and added noise with $\sigma = 0.00$, 0.05, 0.10 and 0.20 (SNR $= \infty$, 21, 15, and 9 dB). The pseudoinverse and HK both showed a drop in performance P_c around $M \approx N$ (as expected) when noise is present (Figure 7). The noise-free HK gave $P_c \geq 99.9\%$ for $M = 1.52N$ (the standard pseudoinverse only

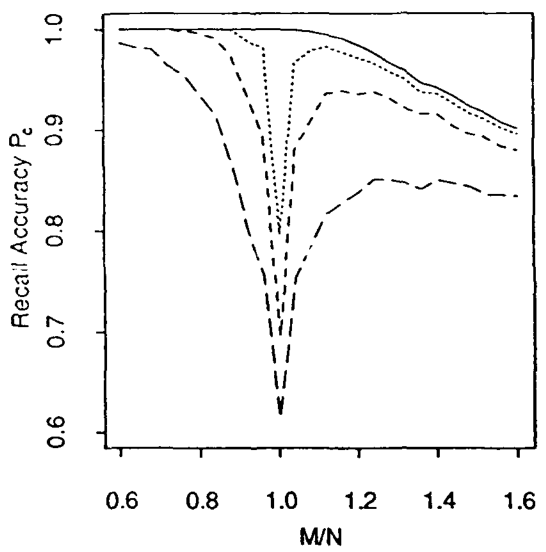


Figure 7: Pseudoinverse 1:1 General HAP results ($N = K = 50, \sigma_n = 0.0, 0.05, 0.10, 0.20$)

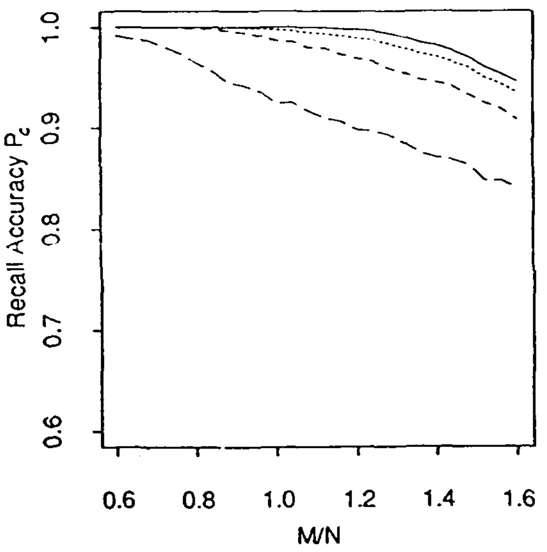


Figure 8: HK-2 1:1 General HAP results ($N = K = 50, \sigma_n = 0.0, 0.05, 0.10, 0.20$)

allowed $M = 1.04N$). We showed earlier [13] that the HK has a maximum storage (general case) of $M = 2N$. This storage of $M = 80$ key/recollection vector pairs with $P_c \geq 99.9\%$ in a 50×50 HAP matrix is much better than what standard correlation HAPs provide ($M \approx 2$). Figure 7 shows the pseudoinverse HAP results. Figure 8 shows the HK-2 robust results. We see no drop in performance near $M = N$ and excellent results, e.g. $P_c > 90\%$ for $M = 1.6N$ with $\sigma_n = 0.1$ (storage of 80 key/recollection 50-element vector pairs in a $50 \times 50 = 2500$ element matrix) and $P_c > 99\%$ for $M = 0.6N$ with $\sigma_n = 0.2$ (25 vector pairs (50 element vectors) in a 2500 element matrix).

4.5.2 Comparison of Different HAP Algorithms

Table 5 compares the performance of different HAPs with $K = 7$ binary (± 1) encoding. With $K = 7$, we allow $2^7 = 128$ key vector pairs and satisfy $M \geq 2N = 100$ (the maximum theoretical value). We increased M and calculated the maximum M for which $P_c' \geq 95\%$ for different amounts of noise σ_n. We show M/N with each entry being the average of ten noise runs with the same σ_n. All HAPs are the same size (7×50) except the DSNN (it is $M \times 50$). We include the DSNN for comparison. Its M is largest but its size is also much larger (for $M/N = 1$, the DSNN is seven times the size of the other HAPs). The correlation HAP performs worst as expected. In this 1:1 (Table 5) vs. many:1 (Table 3) case, the correlation HAP storage is much worse. If we had used $N = K = 50$, we would expect $M = N/(4log_2N) = 2$ for random binary recollections. Thus, we see that binary encoding improves performance ($M \approx 0.3N = 15$). We find that HK is better than the pseudoinverse (both give poor noise performance for $M \approx N$ and hence low M/N). The robust algorithms improve performance (HK-2 is better than HK and the Pseudoinverse-2 is better than the Pseudoinverse) in noise.

We used $\sigma = 0.1$ in the synthesis of all robust algorithms. We note a drop in P_c' for $\sigma > 0.1$ with HK-2 giving the best performance for $\sigma \leq 0.1$. These Tables 5 and 6 data use a new matrix (different M) for each σ and algorithm (column) test with each entry being the average of ten runs (same σ_n). These entries are obtained from curves as in Figures 7 and 8 and tabulated as shown. HK-2 has M/N always equal or better than that for the Pseudoinverse-2. It appears that the HK-2 algorithm is only better for low σ. To address this, we formed both robust HAPs with $\sigma = 0.2$. Table 7 shows our test results. As seen, HK-2 is better and never worst but it is only appreciably better for σ_n less than the σ value used in the synthesis algorithm.

σ_n	DSNN	Correlation	Standard Pseudoinverse	Standard HK	Robust-2 Pseudoinverse	Robust-2 HK
0.00	> 20	0.36	1.20	> 1.60	1.12	> 1.60
0.05	> 20	0.32	0.92	0.92	1.08	1.32
0.10	> 20	0.24	0.80	0.84	0.92	0.92
0.20	> 20	0.20	< 0.60	< 0.60	< 0.60	< 0.60

Table 5: Different 1:1 General HAPs with $N = 50$ element analog uniformly distributed [-1,1] keys and binary encoded recollection vectors (except for DSNN). The maximum M/N for $P_c' = 95\%$ performance in noise ($\sigma_n = 0.00, 0.05, 0.10, \text{and } 0.20$) is given.

4.6 1:1 General Large-Class PR HAP (Recollection Vector Choices)

We now consider how different recollection vector choices affect the performance of the best 1:1 General PR HAP (HK-2 with $\sigma = 0.1$) with $N = 50$ element analog uniformly distributed keys and with four different output recollection vector encoding choices and different noise σ_n for three different P_c'. As before, we increased M until P_c' (note we use P_c' not P_c) decreased below $P_c' = 90\%$, 95% and 99%. Table 6 lists the results (the maximum M/N) for various cases. The four output recollection vector encodings used were: binary (we used a fixed $K = 7$ bits, since $2^7 = 128$ and $M = 2N = 100$ is the maximum HK capacity), Hamming (we used a $K = 11$ bit word, 7 bits of data and 4 error correction bits to correct one data bit error, with a matrix-vector product postprocessing step), BCH coding ($K = 15$ bits, 7 data bits; this corrects 2 data bit errors and requires iterative matrix-vector postprocessing to achieve this) and L-max coding (with $L = 2$ ones in a $K = 14$ bit output and $L = 3$ ones in a $K = 9$ digit output). We find L-max coding ($L = 2$, $K = 14$ bits) to be preferable. Since the HAP size is small here (7×50 for binary and 14×50 for L-max) we allow a doubling of matrix size to obtain the increased storage ($M = 0.92N$ for binary vs. $M = 1.4N$ for L-max for $P_c' = 95\%$). Doubling K (7 in binary to 14 in L-max) gives a 50% to 100% improvement in M/N. We note that increasing K for binary encoding will not increase M.

We do not use Hamming or BCH since they require added matrix-vector postprocessing. Comparing the Hamming and BCH data to the binary results, we see the improvement this matrix-vector error correction postprocessing provides. We note that L-max encoding also requires postprocessing (finding the L largest outputs). This can be done by running WTA L times or more easily by modifying the WTA algorithm (using a ramp threshold). Thus, the WTA postprocessing is much easier than others (Hamming postprocessing is also quite easy, but this corrects only one error bit). As the required P_c' increases, M/N for the error correcting encodings drop while L-max encoding maintains performance (since error correcting can only correct one and two bit errors). Earlier, we noted that unit vector encoding with WTA is better than binary. Here, modified WTA (L-max) is also better. This is necessary in 1:1 cases to compete with the DSNN.

From the $P_c' = 95\%$ data, we note no appreciable drop in P_c' up to $\sigma = 0.1$ (the value used in synthesis). As noise increases, we note that M/N for binary encoding decreases the most.

Acknowledgements

The support of this work by a grant from AFOSR is gratefully acknowledged.

P'_c (%)	σ_n	Recollection Vector Encoding (7 Data Bits)				
		Binary	Hamming	BCH	L-max (L ones)	
		($K = 7$)	($K = 11$)	($K = 15$)	$K = 14(L = 2)$	$K = 9(L = 3)$
90	0.00	> 1.60	1.56	> 1.60	> 1.60	> 1.60
	0.05	> 1.60	1.52	> 1.60	> 1.60	> 1.60
	0.10	0.96	1.52	1.48	> 1.60	1.44
	0.20	0.68	0.84	0.92	1.12	0.80
95	0.00	> 1.60	1.48	1.56	> 1.60	> 1.60
	0.05	1.32	1.36	1.52	> 1.60	1.44
	0.10	0.92	1.12	1.40	1.40	1.16
	0.20	< 0.60	0.72	0.80	0.96	0.72
99	0.00	1.56	1.32	1.44	1.48	1.36
	0.05	0.88	1.20	1.32	1.40	1.20
	0.10	0.76	0.92	1.08	1.20	0.88
	0.20	< 0.60	0.60	0.64	0.72	< 0.60

Table 6: Different recollection vector encodings for our 1:1 General Robust-2 HK HAP with $\sigma = 0.1$ and $N = 50$ element analog uniformly distributed [-1,1] keys. The maximum M/N for $P'_c = 90\%, 95\%, 99\%$ performance in noise ($\sigma_n = 0.00, 0.05, 0.10, \text{and } 0.20$) is given.

P'_c	σ_n	HK-2	Pseudoinv-2	P'_c	σ_n	HK-2	Pseudoinv-2
90%	0.00	> 1.60	1.20	95%	0.00	1.36	1.04
	0.05	1.48	1.08		0.05	1.20	1.00
	0.10	1.08	0.96		0.10	0.88	0.88
	0.20	0.72	0.72		0.20	0.64	0.60

Table 7: HK-2 and Pseudoinverse-2 Performance with $\sigma = 0.2$ in Synthesis

References

[1] D. Casasent and E. Botha, "A symbolic neural net production system: Obstacle avoidance, navigation, shift invariance and multiple objects," *Proc. SPIE*, vol. 1195, pp. 280-290, November 1989.

[2] D. Casasent, E. Botha, J. Wang, and R. Ye, "Optical laboratory realization of a symbolic production system," *Proc. SPIE*, vol. 1295, April 1990.

[3] P. de Groot and R. Noll, "Reconfigurable bipolar analog optical crossbar switch," *Applied Optics*, vol. 28, pp. 1582-1587, 15 April 1989.

[4] B. Telfer and D. Casasent, "Updating optical pseudoinverse associative memories," *Applied Optics*, vol. 28, pp. 2518-28, 1 July 1989.

[5] B. Telfer and D. Casasent, "Ho-Kashyap optical associative processors," *Applied Optics*, vol. 29, pp. 1191-1202, 10 March 1990.

[6] B. Telfer and D. Casasent, "Ho-Kashyap content-addressable associative processors," *accepted for June 1990 IJCNN.*

[7] D. Casasent and B. Telfer, "Key and recollection vector effects on heteroassociative memory performance," *Applied Optics*, vol. 28, pp. 272-83, 15 Jan. 1989.

[8] D. Casasent, "Optical morphological processors," *Proc. SPIE*, vol. 1350, July 1990.

[9] S. Liu and D. Casasent, "Iterative fisher/minimum-variance optical classifier," *Pattern Recognition*, vol. 23, no. 3/4, pp. 385-391, 1990.

[10] J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proc. Natl. Acad. Sci. USA*, vol. 79, pp. 2554-58, April 1982.

[11] J. Hopfield, "Neurons with graded response have collective computational properties like those of two-state neurons," *Proc. Nat. Acad. Sci. USA*, vol. 81, pp. 3088-3092, 1984.

[12] B. Montgomery and B. Kumar, "An evaluation of the use of the Hopfield neural network model as a nearest-neighbor algorithm," *Applied Optics*, vol. 25, pp. 3759-66, 15 Nov. 1986.

[13] B. Telfer and D. Casasent, "Ho-Kashyap associative processors," *Proc. SPIE*, vol. 1005, pp. 77-87, Nov. 1988.

[14] G. Lendaris and G. Stanley, "Diffraction-pattern sampling for automatic target recognition," *Proc. IEEE*, vol. 58, pp. 198-205, 1979.

CHAPTER 4

"Minimum-Cost Ho-Kashyap Associative Processor for Piecewise-Hyperspherical Classification"

IJCNN '91
SEATTLE
(July)

Minimum-Cost Ho-Kashyap Associative Processor for Piecewise-Hyperspherical Classification

Brian Telfer[1] and David Casasent
Center for Excellence in Optical Data Processing
Department of Electrical and Computer Engineering
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

A new synthesis algorithm is presented for generating a neural associative processor with piecewise-hyperspherical decision boundaries. Two important characteristics of the algorithm are that it represents each class with a near-minimum number of hyperspheres and has proven convergence properties. Classification results are presented for a three-class 3-D distortion-invariant aircraft case study (invariant to changes in position, scale, and in-plane and out-of-plane rotation).

1 Introduction

Associative processors (APs) are attractive for pattern recognition (PR) because they associate noisy and distorted versions of an input key vector with the same class label, or output recollection vector. However, simply assigning prechosen recollections can yield poor recall accuracy. Therefore, we have extended the Ho-Kashyap (HK) [1] AP algorithm [2] to compute recollection vectors that represent arbitrarily complex class boundaries with multiple hyperspheres. Section 2 reviews existing piecewise-hyperspherical classifiers and advances our new minimum-cost HK AP. Section 3 presents classification results for a simple two-class synthetic PR problem with a 2-D feature space. Classification results from our new AP and other classifiers are given in Section 4 for the three-class 3-D distortion-invariant (DI) aircraft case study.

2 Piecewise-Hyperspherical Classification

Others [3,4] have developed piecewise-hyperspherical classifiers, but they do not necessarily find a near-minimum number of hyperspheres or are not guaranteed to converge [3] as ours is.

2.1 General Approach

Denoting the M keys and recollections as the vectors $\mathbf{x}_m$ (N-dimensional) and $\mathbf{y}_m$ (K-dimensional), the vectors $\mathbf{x}_m$ and $\mathbf{y}_m$ form an associated key/recollection pair . The recollections are bipolar binary class labels which the synthesis algorithm determines. We desire a $K \times N$ matrix M satisfying $\mathbf{y}_m = sgn(\mathbf{M}\mathbf{x}_m)$ for $m = 1, \ldots, M$, where sgn is the signum function. Defining matrices X ($N \times M$) and Y ($K \times M$) with the key and recollection vectors as their columns, we must satisfy $sgn(\mathbf{MX}) = \mathbf{Y}$. We synthesize each row $\mathbf{m}_k^T$, $k = 1, ..., K$, of the memory matrix sequentially so that the $\mathbf{m}_k^T$ satisfy

$$sgn \left\{ \begin{bmatrix} \mathbf{m}_1^T \\ \cdot \\ \cdot \\ \mathbf{m}_K^T \end{bmatrix} \mathbf{X} \right\} = \begin{bmatrix} \mathbf{b}_1^T \\ \cdot \\ \cdot \\ \mathbf{b}_K^T \end{bmatrix}, \tag{1}$$

where the $\mathbf{b}_k^T$ are the rows of the bipolar binary recollection matrix $\mathbf{Y}$, i.e., the desired outputs for the $\mathbf{m}_k^T$. To define hyperspherical boundaries, we augment each key in the original N'-dimensional feature space with one element that is 1 and one element that is the sum of the squared original N' key elements [5]. This creates new keys of dimension $N = N' + 2$. The $\mathbf{m}_k^T$ define hyperplanes in the new N-D (N-dimensional) space. However, these hyperplanes are hyperspheres in the original N'-D space.

To calculate $\mathbf{m}_1^T$ in (1), we treat one class as the true class and all others as the false class. The elements of the desired output $\mathbf{b}_1^T$ are specified to be +1 for true class keys and -1 for false class keys. We solve for an m_1^T that produces a +1 output for many true class keys and a -1 output for all or almost all false class keys.

[1]now at the Naval Surface Warfare Center, Silver Spring MD, supported by an ONR Young Navy Scientist Award.

M_T/M_F	0-0.01	0.01-0.02	0.02-0.05	0.05-0.1	0.1-0.2	0.2-0.5
c	1	1.5	2	2.5	3	3.5

Table 1: Guidelines for false class costs c for different M_T/M_F (the ratio of the remaining true class keys to the false class keys).

Our primary concern is that no (or very few) false class keys have +1 outputs. The hypersphere defined by $\mathbf{m}_1^T$ thus correctly classifies some number of the true class keys and separates them from almost all of the false class keys. We then remove the correctly classified true class keys from **X** (and the associated elements from $\mathbf{b}^T$) and calculate an $\mathbf{m}_2^T$ that correctly classifies (+1 output) some of the remaining true class keys and rejects (-1 output) almost all false class keys. This procedure continues. When all $\mathbf{m}_k^T$ that classify this class and reject all others have been calculated, we select a second class as the true class and assign all others to the false class, etc. and we continue calculating $\mathbf{m}_k^T$ rows until hyperspheres to separate each class from all others have been determined. Our approach avoids overtraining (because hyperspheres that separate only a few true class keys from the false class keys need not be used), uses few hyperspheres and reasonable computer time.

The recall operation is $\mathbf{y} = sgn(\mathbf{Mx})$. The k-th output element y_k indicates whether the input **x** falls on the positive or negative side of the k-th hypersphere. If the input falls on the positive side of any hypersphere from a class, and falls on the negative side of all hyperspheres from other classes, then the input is classified as that class. If the input falls on the positive side of hyperspheres from more than one class, then the input is rejected (i.e. no decision is made). The input is also rejected if the input falls on the negative side of all hyperspheres. The reject capability is important. It is not always needed, since the input can be classified according to the maximum output element.

2.2 Minimum-Cost HK Algorithm

We find the $\mathbf{m}_k$ in (1) by minimizing the criterion function $J = (\mathbf{X}^T\mathbf{m}_k - \mathbf{b}_k)^T\mathbf{C}(\mathbf{X}^T\mathbf{m}_k - \mathbf{b}_k)$, where **C** is a diagonal cost matrix containing positive weights. Depending on k, **X** may not contain the entire set of true class keys, i.e. when calculating subsequent $\mathbf{m}_k$ for one class, only keys not classified by a previously computed $\mathbf{m}_k$ are included as noted in Section 2.1. The term $(\mathbf{X}^T\mathbf{m}_k - \mathbf{b}_k)$ is a vector of errors between the actual $\mathbf{X}^T\mathbf{m}$ and desired **b** outputs. The criterion function J is the sum of the squared errors with each error weighted by an element in **C**. When $\mathbf{C} = \mathbf{I}$, **M** is our standard HK AP [2]. We use **C** to heavily weight the errors for the false class keys to ensure that few or none are misclassified. This gives low error rates, good recall accuracy, and a small number of $\mathbf{m}_k$.

We first discuss how to select the proper costs. This requires a few iterations. We fix the cost for true class errors at 1 and vary the false class cost during the cost iterations. In our aircraft case study (Section 4), we define a satisfactory cost to be a value that results in a hypersphere with a number of false class errors less than 5% of the number of correct true class keys. The 5% figure can be varied (allowing higher percentages will decrease the number of hyperspheres, but will decrease the recall accuracy; requiring a lower percentage will increase recall accuracy, but will increase the number of hyperspheres). In our 2-D two-class example (Section 3), we use 0% instead of 5%, for demonstration purposes only (0% is unrealistic for the complex PR problems for which our minimum-cost APs are intended).

We now consider choosing the initial and subsequent false class costs. The false class cost used should decrease as true class keys are removed from **X** when finding a sequence of hyperspheres, since the sum of squared errors is increasingly dominated by false class errors. From our aircraft case study (Section 4), we have devised guidelines (Table 1) for the choice of initial false class costs as a function of M_T/M_F (the ratio of the remaining true class keys to the false class keys). If the initial cost is too low (i.e. if it produces too many false class errors), we multiply the cost by 3/2. If the initial cost is too high (i.e. if there are almost no false class errors but many true class errors), we multiply the cost by 2/3. If upper and lower limits for the cost exist (from two or more cost iterations), then we choose a new cost that is midway between the upper and lower limits. Using these rules on our aircraft database, each hypersphere only requires 1.8

cost iterations to determine a satisfactory $\mathbf{m}_k$. We expect these rules to apply to other pattern recognition problems, although more cost iterations may be required These rules are not critical to the minimum-cost AP synthesis algorithm, i.e. the rules could be changed and we would still find satisfactory costs. The critical part of the minimum-cost AP synthesis is the minimum-cost algorithm itself, which we now discuss. It has a strong analytical basis and proven convergence properties.

We now derive our minimum-cost AP. Setting the gradient of J with respect to $\mathbf{m}_k$ equal to zero and solving for $\mathbf{m}_k$ gives the new minimum-cost pseudoinverse AP solution to $\mathbf{m}_k^T \mathbf{X} = \mathbf{b}_k^T$ in (1) as $\mathbf{m}_k = \mathbf{X}_{MC}^{+T}\mathbf{b}_k$ where $\mathbf{X}_{MC}^{+} = \mathbf{C}\mathbf{X}^T(\mathbf{X}\mathbf{C}\mathbf{X}^T)^{-1}$ is the minimum-cost pseudoinverse. The idea behind the HK algorithm is to refine the pseudoinverse AP by iteratively changing the $\mathbf{b}$ elements, but not their signs, in order to reduce J and thereby reduce the classification error. The gradient descent algorithm for the minimum-cost HK AP is obtained by taking the gradient of J with respect to $\mathbf{m}_k$ <u>and</u> $\mathbf{b}_k$. Eq. (2) gives the algorithm. In (2), we omit the k subscripts on $\mathbf{m}$ and $\mathbf{b}$ for simplicity and use subscripts (i) to denote the iteration number. The algorithm operates in the same manner as the basic HK algorithm, except that it uses a new method for forming the modified error $\mathbf{e}'$ and it incorporates the cost matrix. We now discuss this algorithm.

We begin with an estimate of $\mathbf{m}$ (step 1). We modify $\mathbf{b}$ (step 4) and $\mathbf{m}$ (step 1) in successive iterations (index i). In step 2, we calculate the error vector $\mathbf{e}$. In step 3, we form a modified error vector $\mathbf{e}'$ with elements e'^m (we use $\epsilon = 0.1$). In Step 4, we add $\mathbf{e}'$ to $\mathbf{b}$ to alter the recollection elements b^m of $\mathbf{b}$. The $\mathbf{e}'$ vector equals $\mathbf{e}$ except that any $\mathbf{e}'$ elements that would cause the desired outputs $\mathbf{b}$ to change sign (or to come within ϵ of changing sign) are set to zero. The $\mathbf{e}'$ vector differs from the one used in the standard HK algorithm in the use of ϵ and in how the b^m elements are altered. The standard HK algorithm only allows the b^m magnitudes to increase, while our method in Step 3 allows the b^m elements to increase or decrease. This minimizes J faster than the standard method, and therefore requires 20% fewer HK synthesis iterations i in the Section 4 case study.

Step	Operation
1	$\mathbf{m}_i = \mathbf{X}_{MC}^{+T}\mathbf{b}_i$
2	$\mathbf{e}_i = \mathbf{C}^{\frac{1}{2}}(\mathbf{X}^T\mathbf{m}_i - \mathbf{b}_i)$
3	if $b_i^m > 0$, $e_i'^m = max\{(\epsilon - b_i^m)/(\rho C_{mm}^{1/2}), e_i^m\}$
	else $e_i'^m = min\{-(\epsilon + b_i^m)/(\rho C_{mm}^{1/2}), e_i^m\}$ (2)
4	$\mathbf{b}_{i+1} = \mathbf{b}_i + \rho\mathbf{C}^{\frac{1}{2}}\mathbf{e}_i'$,
5	If $\mathbf{e}_i' \neq 0$ go to 1.

We have proved that the algorithm converges when $0 < \rho < min(2, 2/c)$, with $\rho = min(2, 2/c)$ giving the fastest convergence. If the two classes are linearly separable in N dimensions (or equivalently, hyperspherically separable in $N' = N-2$ dimensions), the algorithm will find an LDF that separates them. If the classes are not linearly separable (as will usually be the case), then the algorithm will find an LDF that minimizes J. The proof is an important result, because it provides a mathematical basis for the algorithm. In practice, we stop the HK algorithm when all keys are correctly classified or when the number of misclassified keys is unchanged after 50 iterations.

After computing one or several hyperspheres, the algorithm may not be able to generate a hypersphere that correctly classifies a reasonable number of the remaining true class keys (we use one-fourth as our rule) without incorrectly classifying many false class keys. When this occurs, we use the basic Isodata clustering algorithm [6] to divide the remaining true class keys into two clusters, and then apply the minimum-cost HK algorithm to each cluster separately. For difficult PR problems (for which this algorithm is intended) such as the aircraft case study, we cluster after each hypersphere is computed.

3 2-D Example

In this section we present results of our minimum-cost HK algorithm on two classes of 2-D feature vectors. This is done to provide a simple example of our procedure for a case in which the class boundaries can be easily visualized. Figure 1 shows two class distributions with piecewise-circular decision boundaries found

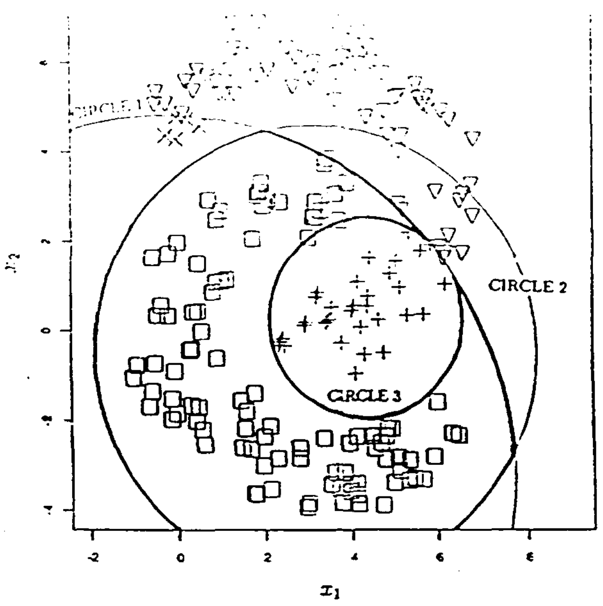


Figure 1: Piecewise-circular boundary found between two 2-D classes by the minimum-cost HK AP.

with the minimum-cost HK algorithm. The false class training vectors are depicted by squares, and the true class training vectors are plotted as triangles, x's and crosses, depending on which LDF correctly classified them. Since this example has only two classes, we compute circles for only one class as the true class (this does not allow for a reject decision). To find circular boundaries, we augment the 2-D feature vectors with two elements as explained in Section 2.1. The minimum-cost HK algorithm sequentially computes three LDFs describing the three circular boundaries in Figure 1. The final boundary between the two classes is marked with a heavy line. The resulting minimum-cost HK AP has a 3×4 memory matrix, with three output elements (one for each circle) and four input elements (the original two elements plus two additional ones). During recall, if any output element is positive, the input is classified as the true class. Otherwise, the input is classified as the false class. Any inputs from the true class side of the heavy line in Figure 1 have at least one positive output element. Any inputs from the false class side of the heavy line have all negative output elements. The training set is classified with 100% accuracy. We have not included a test set because our purpose is to give insight into how the class boundaries are generated.

We now detail how the minimum-cost HK algorithm found the three LDFs. We fixed the true class costs at 1, and varied the false class costs to correctly classify all of the false class (squares) while correctly classifying as many keys from the true class as possible. (In this two-class 2-D example, we attempt to correctly classify all (not most) false class keys. In practical and complex applications (e.g. Section 4), we find hyperspheres in which the number of false class errors does not exceed 5% of the number of true class keys correctly classified.) After finding circle 1 and removing the correctly classified true class keys, we were unable to find a false class cost that would correctly classify all of the false class keys and more than one-fourth of the true class keys. This occurs because, as seen from Figure 1, the remaining true class keys (the x's and the crosses) form two clusters on opposite sides of the false class distribution. We therefore use the Isodata clustering algorithm to separate the true class keys into two clusters. Applying the minimum-cost HK algorithm to the two clusters and to the false class keys produced circles 2 and 3 which correctly separated the x's as one cluster and the crosses as another. For this example, we tried an average of three costs per circle. Because the minimum-cost HK algorithm requires only about 78 iterations per hypersphere (one minute) for this 2-D example, we made little effort to minimize the number of costs tried.

4 3-D DI Aircraft Case Study

We consider three classes (DC10, F4, F104) of 128×128 aircraft imagery. As a key vector representation space, we use 32 wedge samples of the Fourier transform intensity (in half of the transform plane). The wedge feature space provides scale invariance (when the wedge samples are normalized) and shift invariance,

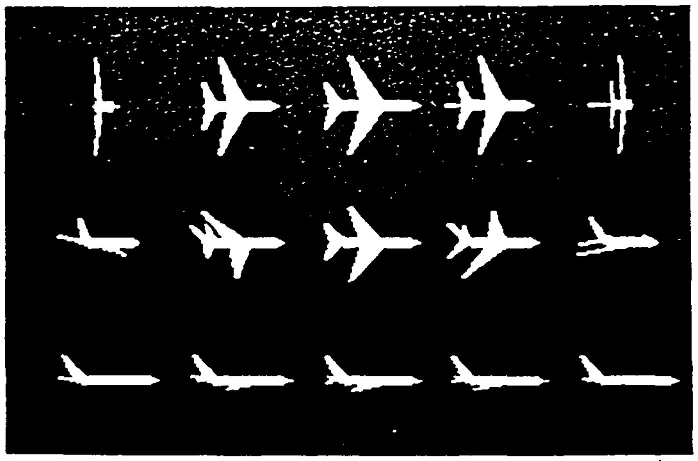


Figure 2: Representative roll (vertical 0°, −40°, −80°) and pitch (horizontal, 0°, ±40°, ±80°) views of DC10.

	Training P_c' (%)	Test Set P_c' (%)	K
DSNN	100	99	3267
Min-Cost HK-A	98.9	98.1	24
Min-Cost HK-B	97.4	97.1	18
Min-Cost Pseudoinverse	90.9	91.5	47
ACNN	91.4	92.3	24
Standard Linear HK	80.7	81.2	3

Table 2: Classification accuracies for the three class aircraft case study for six different methods.

and is easily generated optically [7]. In-plane image rotations cause the wedge samples to circularly shift. We postulate that the aircraft are moving and that tracking information provides the location of the aircraft's nose. This allows the wedge samples from an unidentified aircraft to be circularly shifted so that they align properly with the training vectors. Thus, for moving aircraft this feature space is rotation (in-plane), scale and shift invariant. We use a training set of 3267 key vectors produced from the three aircraft rotated out-of-plane in pitch and roll between ±80° at 5° increments. Figure 2 shows representative views of the DC10 to show the wide range of out-of-plane views we consider. The test set contains 3072 key vectors (not present in the training set) produced from the three aircraft rotated in pitch and roll between ±77.5° at 5° increments (i.e. at least 2.5° different in pitch and roll from the training data).

Table 2 gives classification accuracies for six different APs, with P_c' defined to be the percentage of input vectors correctly recalled. The five classification methods considered include the Direct Storage Nearest Neighbor (DSNN) AP [8] (a nearest neighbor classifier expressed in AP matrix form), the minimum-cost pseudoinverse and HK APs, the Adaptive Clustering Neural Net (ACNN) [9] and a standard HK AP. All five methods employ maximum selection of the output vector elements. (We consider a threshold for the minimum-cost APs later.) For each of these APs, $N = 33$. There are 32 elements due to wedge features and an additional element for each AP. For the minimum-cost APs, we require two additional elements but, because the keys are normalized, the second additional element that equals the sum of squared key elements always equals one and is therefore omitted. For the other APs, the additional element varies. The size of each AP is thus $K \times 33$, where K is given in Table 2.

We now consider the classification results of these APs. Although the DSNN AP is too large to be practical ($K = 3267$), its excellent classification results indicate that the wedge feature space does separate the different classes into distinct clusters. Our minimum-cost HK APs (A and B) use 24 and 18 hyperspheres, which reduce the number of weight vectors by two orders of magnitude over the DSNN AP while still retaining

	Threshold			Max Selection	
	P'_c (%)	Reject (%)	P'_e (%)	P'_c (%)	P'_e (%)
Training Set	93.9	5.7	0.4	98.9	1.1
Test Set-1	92.5	6.8	0.7	98.1	1.9

Table 3: Classification accuracies for minimum-cost HK-A AP using thresholding.

excellent recall accuracy. The minimum-cost HK-A AP was computed according to the rules described in Section 2. The AP synthesis time was about 75 CPU minutes on a Sun Sparcstation. An average of only two trials for each hypersphere was needed to determine an acceptable cost. Running the HK routine required an average of 243 iterations. The minimum-cost HK-B AP was created by eliminating the six hyperspheres in the HK-A AP containing the fewest training vectors (these hyperspheres are known from the original training procedure). This is an example of the flexibility of the minimum-cost AP in adjusting the number of hyperspheres (as discussed in Section 2.1). The smaller HK-B AP memory matrix is achieved at the expense of a slightly lower P'_c. The minimum-cost HK APs are clearly preferable to the minimum-cost pseudoinverse AP, which requires twice the number of output elements in the HK-A AP while yielding 6.6% more errors in the test set. The ACNN used performs 5.8% worse than the minimum-cost HK-A AP, even though both classifiers use the same number of weight vectors ($K = 24$). We also used the standard HK AP algorithm with conventional three-element unit vectors to denote each class. This gives a small AP with $K = 3$, but significantly worse performance than the other APs.

The results in Table 2 for the minimum-cost APs were found by detecting the maximum output element, rather than thresholding the output. Using a threshold for the minimum-cost HK-A AP gave the results shown in Table 3 (with P'_e defined to be the percentage of input vectors incorrectly recalled). These show a large reduction (about 60%) in the test set errors due to the presence of a reject class. The reduction in P'_e is obtained at the cost of about 5% lower P'_c. The DSNN and ACNN methods could be modified to provide a reject class, but we did not consider this.

In conclusion, our new AP gives 98% accuracy (comparable to the DSNN, but with a storage density that is two orders of magnitude better). The minimum-cost HK AP is attractive for reasons in addition to its good performance and small matrix size. First, it automatically determines how many hyperspheres are needed to represent each class, and hyperspheres containing only a small number of keys can be discarded. Second, our AP offers a rejection capability that has rarely been addressed. Our minimum-cost APs also function well with no rejection (detecting the maximum output element), if that is desired.

Acknowledgement

The support of this research by the Air Force Office of Scientific Research is gratefully acknowledged.

References

[1] Y.-C. Ho and R. Kashyap, IEEE Trans. Elec. Comp. **EC-14**, 683-88 (1965).
[2] B. Telfer and D. Casasent., App. Opt. **29**, 1191-1202 (1990).
[3] B. Batchelor, *Practical Approach to Pattern Classification*, (Plenum, New York, 1974).
[4] D. Reilly, L. Cooper, and C. Elbaum, Biological Cybernetics **45**, 35-41 (1982).
[5] T. Cover, IEEE Trans. Elec. Comp. **EC-14**, 326-34 (1965).
[6] R. Duda and P. Hart, *Pattern Classification and Scene Analysis*, (John Wiley and Sons, NY, 1973).
[7] G. Lendaris and G. Stanley, Proc. IEEE **58**, 198-205 (1979).
[8] B. Montgomery and B.V.K.V. Kumar, App. Opt. **25**, 3759-66 (1986).
[9] D. Casasent and E. Barnard, App. Opt. **29**, 2603-2615 (1990).

CHAPTER 5

"MSE and Hierarchical Optical Associative Processor System"

Proc. SPIE, Vol. 1382
November 1990, Boston

"MSE and Hierarchical Optical Associative Processor System"

DAVID CASASENT and SUNG-IL CHIEN*

Carnegie Mellon University
Center for Excellence in Optical Data Processing
Department of Electrical and Computer Engineering
Pittsburgh, PA 15213

*[Department of Electronics, Kyungbook University, Republic of Korea]

ABSTRACT

We consider high capacity mean square error (MSE) associative processors (APs) and the first use of multiple APs. The application considered is multi-class distortion-invariant pattern recognition (PR).

1. INTRODUCTION

APs are attractive solutions for difficult PR problems. Section 2 reviews the three APs we consider. Section 3 describes our multi-class distorted object database and PR problem chosen. Section 4 describes the feature space input AP neuron representation space we use (feature space neurons are needed to reduce the number of neurons required, to provide some distortion invariance and to reduce the training set size needed). Section 5 presents initial results (these show the need for multiple APs). Section 6 advances our multiple AP concept and Section 7 notes our conclusions.

2. ASSOCIATIVE PROCESSORS

Figure 1 shows the AP matrix-vector (M-V) processor considered and the dimensions of the matrix ($\underline{M}$), the input ($\underline{x}$), and output ($\underline{y}$) vectors where $\underline{M}\,\underline{x} = \underline{y}$ and M is the number of stored vector pairs. The solution for $\underline{M}$ must satisfy $\underline{Y} = \underline{M}\,\underline{X}$, where the columns of $\underline{X}$ and $\underline{Y}$ are the $\underline{x}_k$ and $\underline{y}_k$ training vectors. We consider MSE heteroassociative processors, since they allow useful solutions when M > N, which is essential for an AP to be competitive.

We consider 3 APs (Table 1). The first two use the pseudoinverse solution $\underline{M} = \underline{Y}\,\underline{X}^{+}$. For $\underline{M}_I$, $\underline{Y} = \underline{I}$, and $\underline{M}$ is large (M$\times$N) as is $\underline{y}$ (its dimension is K = M). Since the $\underline{y}$ are unit vectors and $\underline{Y}$ is the identity matrix, we denote this AP matrix as $\underline{M}_I$.

The second AP uses a small K = C sized output vector $\underline{y}$ (where the $\underline{y}$ are unit vectors and C is the number of distored object classes that must be identified). Since we only require the class of the object, we can use small $\underline{y}$ and $\underline{M}$ (which we denote by $\underline{M}_U$ for unit vector).

The final AP considered is the DSNN where $\underline{M}$ is the data matrix denoted by $\underline{M}_D$.

3. PR CASE STUDY AND DATABASE

The problem we chose in multi-class PR is the recognition of the class of 10 different aircraft (Figure 2) when multiple 3-D distortions in roll, pitch, and yaw are present (Figure 3). We divide this large problem into 4 cases (Table 2) containing M = 168-432 images (with 28-72 images per class, each being a different distorted view with different ranges for θ_x and θ_y). In Database 1, the 6 aircraft include 2 from each of the 3 groupings in Figure 2. In Database 2, we omit the 747 from Database 1. In Database 3, we add 24 additional 747 images (those aspect views that gave errors in Database 1 tests). Database 3 is nearly in-plane data. Database 4 contains larger 3-D distortions.

4. INPUT NEURON SPACE

The input neuron space for $\underline{x}$ must have various properties. It should be large (N large) but reasonable to allow more storage M. The feature space used must be easily calculated, should maintain the same dimensionality for distorted inputs (this precludes the use of our earlier string-code feature space), should be in-plane distortion-invariant (to reduce the size M of the training set required, and hence to increase storage capacity), and most of all it must be shift-invariant (or M will radically increase).

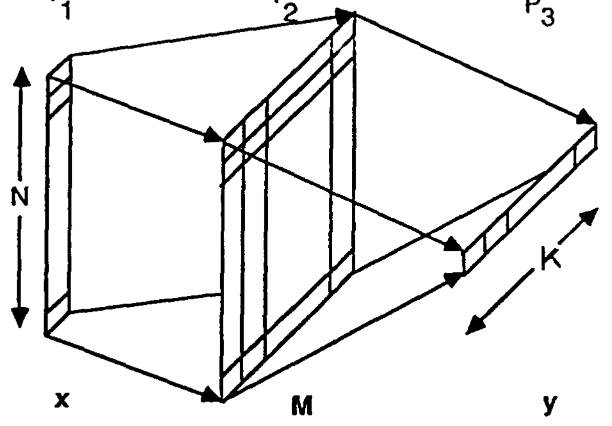


Figure 1: AP rotation/architecture.

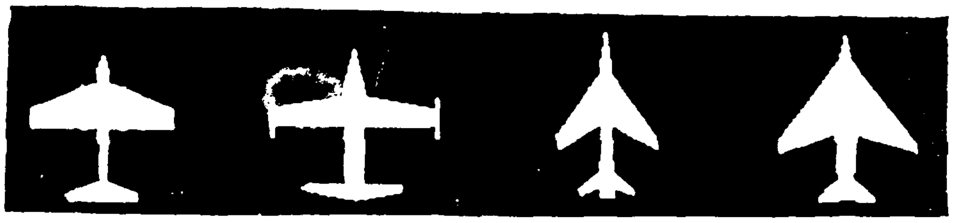
The feature space used was the chord angles ψ and lengths L from the centroid of the object to the object's contour. The angles ψ were chosen to yield equal lengths along the boundary. Figure 4 shows ψ_n and L_n for two contour lines (shown dotted). The centroid was obtained easily from 2 projections. The chords are normalized by the longest L_n. The reference angle ψ_1 is the one associated with the longest chord. This feature space is shift, scale and in-plane rotation invariant and is easily calculated from the Hough transform and (ϕ,s) string code we described earlier. It is similarly robust. Figures 5-8 show an input object and its L, ψ and (L,ψ) feature spaces.

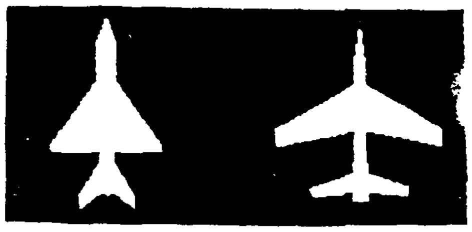
5. INITIAL AP TEST RESULTS

The classification accuracy P_C in Database 1 is shown in Figure 9 for the three APs using 8 values

M_I	OUTPUT IS UNIT VECTOR (M LONG) MATRIX LARGE ($M \times N$) ONE "1" DENOTES WHICH OF M INPUTS PRESENT E.G. C=6 CLASSES, 72 IMAGES OF EACH, M=432 OUTPUT (LOCATION OF "1") DENOTES CLASS AND ORIENTATION FIRST 72 OUTPUTS FOR CLASS 1, NEXT 72 FOR CLASS 2, ETC MX=Y, Y⇒I=IDENTITY MATRIX HAM
M_U	OUTPUT IS UNIT VECTOR (K=C SHORT) SAME UNIT VECTOR FOR EACH CLASS NO ORIENTATION INFORMATION E.G. C=6 CLASSES, OUTPUT VECTOR y OF LENGTH 6 SMALL MEMORY SIZE, UNIT VECTOR HAM
M_D	DATA MATRIX, $M=X^T$ THIS IS M^+ IF x_k ARE OTHONORMAL

Table 1: Three APs considered.


(a) U.S.A. military aircraft: (1) B57 (2) F104 (3) F105 (4) Phantom


(b) Foreign military aircraft: (1) MiG21 (2) Mirage

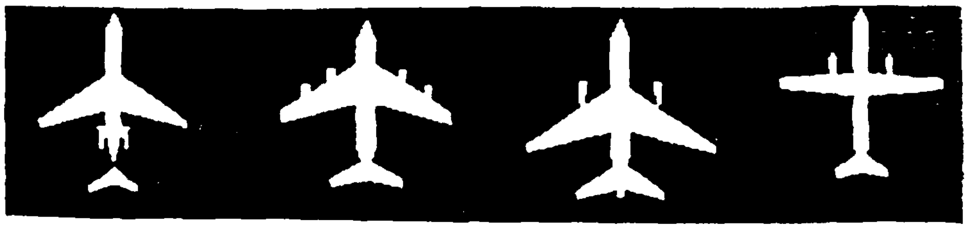

(c) Commercial airliners: (1) B727 (2) B747 (3) DC10 (4) Swearingen

Figure 2: Ten aircraft used in 3 groupings.

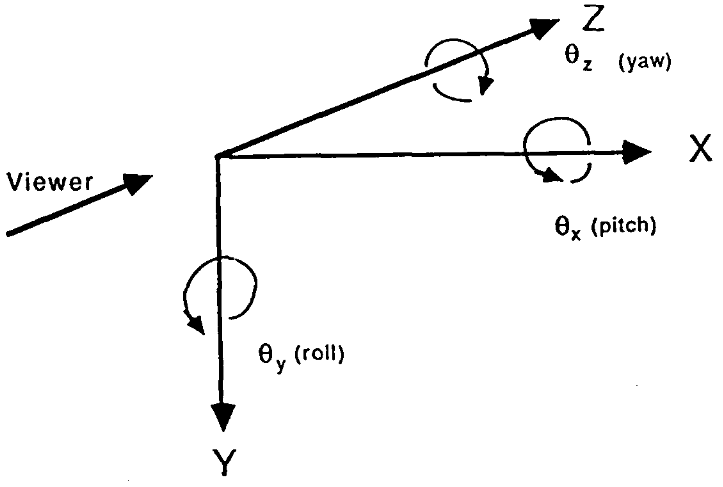


Figure 3: Rotation angles in 3-D

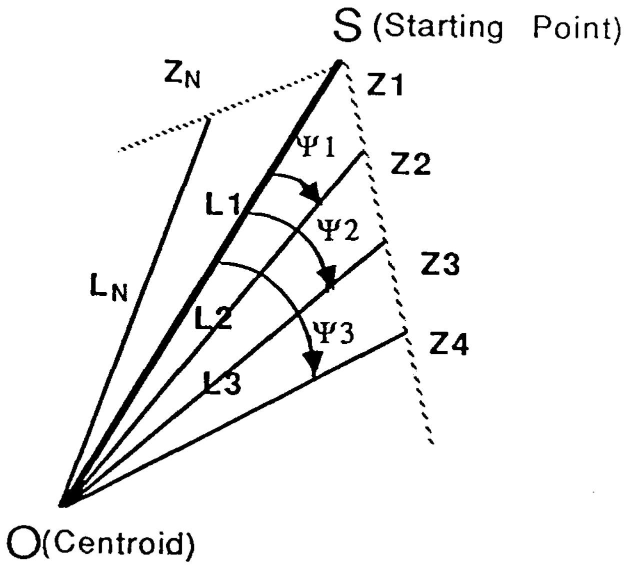


Figure 4: Feature space.

Database	Description	Number of Images M
1	6 aircraft 120° / 10° = 12 angles θ_x and 60° /10 °= 6 angles θ_y	M=432 (72 images per class)
2	5 aircraft (NO B747)	M=360 (72 images per class)
3	6 aircraft 7 angles θ_x (-30° to 30 °) and 4 angles θ_y (0° to 30°)	M=168 (28 images per class)
4	6 aircraft 12 angles θ_x (-60° to 50 °) and 3 angles θ_y (40 ° to 60°)	M=216 (36 images per class)

Table 2: Four databases used.

for N (N = 18, 36, 72, 144, 288, 432, 576 and 864) with M = 432. We show P_C versus N/M (storage capacity). As seen, all APs yield perfect results for $N \geq 2M/3 = 288$ dimension input keys (hence high dimension key vector spaces are most useful). To properly evaluate an AP for distortion invariant PR, we test it on test data (intermediate views at 5° intervals not present in the training set). Figure 10 shows the results obtained. We note that we achieve the best performance for $N/M \simeq 1/3$ or for $N \simeq M/3 = 144$ dimension vectors when M = 432 inputs were stored. We note that reasonable

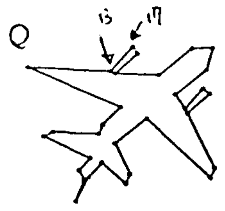


Figure 5: Input DC-10 (θ_x=50°,θ_y=20°, θ_z=60°) for feature space example

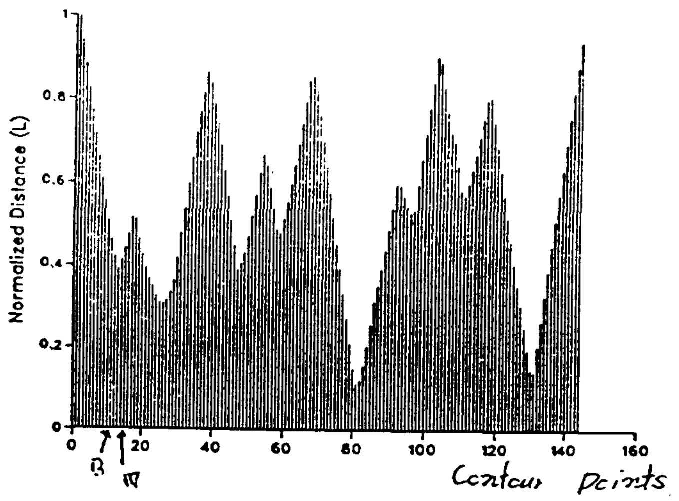


Figure 6: Chord length (L) space
N = 144 chords

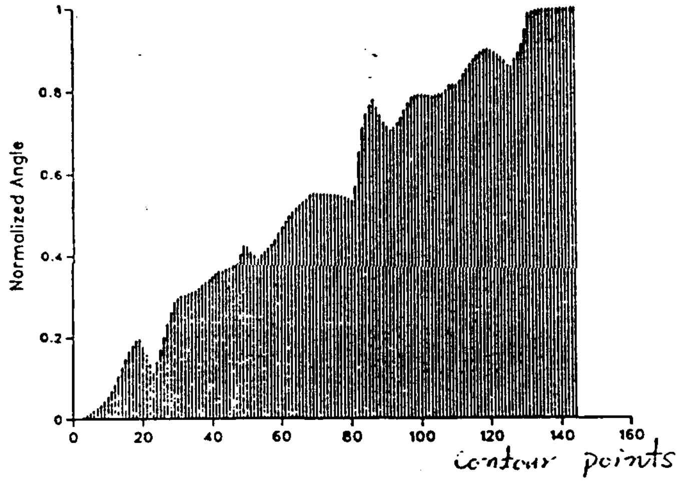


Figure 7: Chord angle (ψ) space
N = 144 chord angles

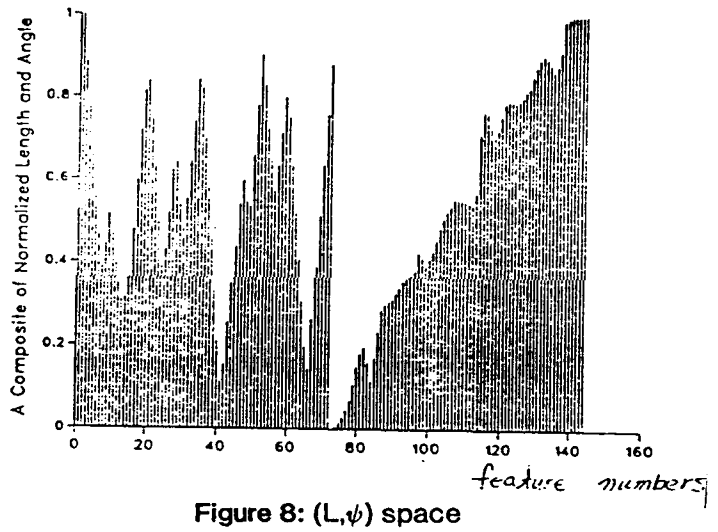


Figure 8: (L,ψ) space

performance is obtained for M_U up to M = 2N (and that performance P_C drops near M $\simeq$ N as is expected).

<u>6. IMPROVED PERFORMANCE AND USE OF MULTIPLE APs</u>

We note that the range of maximum correct largest peak values was large (for M = 432, N = 144 or a large storage of M $\simeq$ 3N) and thus maximum selection (WTA) is required at the output. We also note that the AP with $\underline{M}_U$ is required at the output. We also note that the AP with $\underline{M}_U$ gave a larger minimum than the other APs (<u>thus smaller dimension recollection vectors</u> $\underline{y}$ <u>are preferable</u> as are used in $\underline{M}_U$). Figure 11 shows a typical AP output with one large peak.

We tested these APs for the number of input and matrix levels required (Table 3) and see that the $\underline{M}_U$ associative processor is preferable (for size etc.) as its P_C varies little with as few as 256 levels in the AP matrix and input vector. To improve performance of the $\underline{M}_U$ associative processor, we

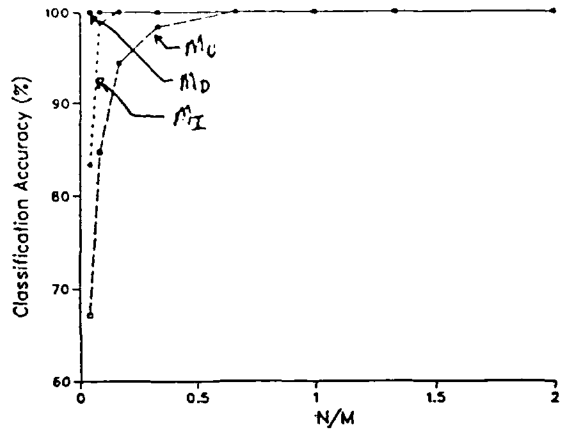


Figure 9: P_C for three APs vs. N/M for Database 1 (training images)

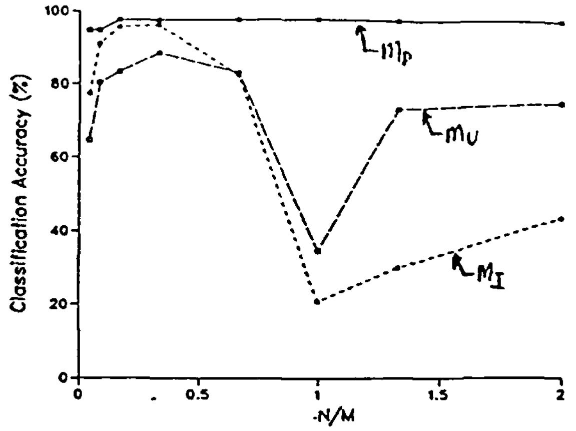


Figure 10: P_C for three APs vs. N/M for Database 2 (test images)

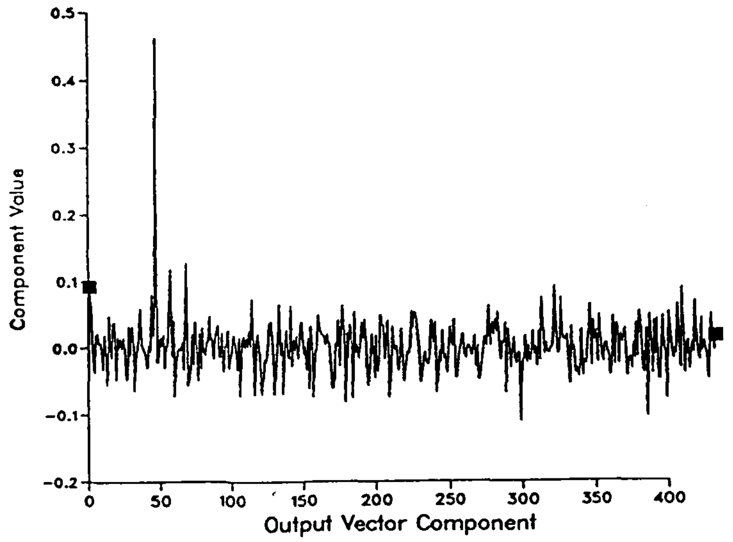


Figure 11: Typical M_I output showing element 47 (DC10 with θ_x=35°, θ_y=35° and θ_z=240°) as the input object

	DC10	F104	MiG21	B747	Phantom	Mirage
DC10	64	0	0	6	2	0
F104	0	63	1	1	3	4
MiG21	1	0	71	0	0	0
B747	0	1	1	52	14	4
Phantom	0	1	0	3	67	1
Mirage	0	3	2	0	2	65

Figure 12: Confusion matrix

formed the confusion matrix (Figure 12) and noted that most errors (20 out of 50) occurred for the B747 and that most other errors (in $\underline{M}_U$ and $\underline{M}_I$) generally occurred for highly distorted inputs (near the maximum θ_x and θ_y).

We thus formed a number of new APs with no 747 images (these are simply too similar to other aircraft) and obtained excellent results (Table 4) with P_C approaching 99.7% for high M/N ratios for both training and test data. This leads to our proposed solution (Figure 13) to such large class (multi-class) pattern recognition problems with severe distortions. This novel approach uses multiple APs (as noted in the second rank(column)). We estimate the orientation of the input (from moments, projections, etc.). We separate the objects into ranges of distortions using separate APs for each (thus making each AP easier and with better performance). The final rank 3 (column 3) APs determine

	Machine Precision	4096 Levels	2048 Levels	1024 Levels	512 Levels	256 Levels	128 Levels	64 Levels
M_I	96.1%	96.1%	94.9%	92.6%	72.2%	36.3%	22.5%	19.9%
M_U	88.4%	88.4%	88.2%	87.7%	88.2%	86.3%	79.2%	43.5%

Table 3: Effects of quantization of matrix and input in our APs.

MEMORY CLASS		RANGE OF PEAKS VALUES	Pc
FULL	TRAINING	0.36-1.26	98.4%
M=432	TESTING	0.29-2.1	88.4%
NO B747	TRAINING	0.4-1.26	99.7%
M=360	TESTING	0.5-1.35	93.3%
IN-PLANE	TRAINING	0.4-1.25	99.4%
M=168	TESTING	0.37-1.5	91.7%
OUT-OF-PLANE	TRAINING	0.41-1.23	99.1%
M=216	TESTING	0.355-1.55	94.0%

Table 4: APs with 747 images omitted (showing much better results).

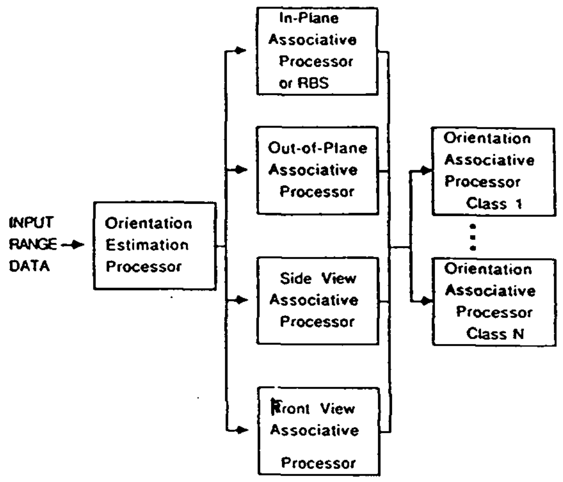

INPUT RANGE DATA
Orientation Estimation Processor
In-Plane Associative Processor or RBS
Out-of-Plane Associative Processor
Side View Associative Processor
Front View Associative Processor
Orientation Associative Processor Class 1
Orientation Associative Processor Class N

Figure 13: Multiple AP concept for large multi-class severe distortion PR.

the orientation and class of the input object. One can extend this concept to determining other class designations (commercial, military, etc.) of the aircraft.

7. SUMMARY

We have shown that large capacity (M > N) APs are possible. We feel that APs with M > N are essential for competitive performance. We found them to give excellent (P_C = 92-99%) performance on a very severely distorted large multi-class distorted object problem and to allow excellent quantization to 128 matrix and input vector levels (allowing analog AP accuracy via analog VLSI or optical realizations).

ACKNOWLEDGMENT

The support of this research by a grant from the Air Force Office of Scientific Research is gratefully appreciated.